commodore Magazine

AÑO I - Núm. 2 - Abril 1984 * 250 Ptas.

REVISTA INDEPENDIENTE PARA USUARIOS



CBM 64 en profundidad

Superbase 64: El ordenador que archiva

Juegos, trucos y aplicaciones

ELLOS SON FAMOSOS EN EL MUNDO DE LOS VIDEOJUEGOS

(BUGABOO. LA PULGA. N.° 1 EN INGLATERRA)

BUGABOO C.B.M. 64

BUGABOO Spectrum

FRED Spectrum

TU JUEGO

¿QUIERES SERLO TU TAMBIEN?

¿Te gusta programar? ¿Eres capaz de escribir en código máquina? ¿Tienes una buena idea? ¿Has conseguido alguna rutina espectacular? ¿Estás creando un juego?..... INDESCOMP. SOFTWARE, invita a todos los entusiastas de la programación con conocimientos y práctica del lenguaje ensamblador (código máquina 6502 y Z80) a participar y colaborar con nuestro Departamento de videojuegos en la creación de nuevos ''best sellers''.

Envianos urgentemente tus ideas y experiencias a:



SOFTWARE

DIRECTOR DE PROGRAMAS
P.° DE LA CASTELLANA, 179 - MADRID-16 - TELEF. 279 31 05

commodore Magazine

Sumario

Commodore Magazine es una publicación de Ediciones y Suscripciones, S. A. C/ Jerez, 3. Madrid-16.
Presidente: Fernando Bolín.
Director Editorial: Norberto Gallego.
Director: Alejandro Diges.
Redacción: (91) 250 15 92.
Aníbal Pardo.
Gumersindo García.
Roberto Menéndez.
Simeón Cruz.
Miguel Angel de Frutos.
Manuel Arias.
Diseño: R. Segura.
Gerente de circulación y ventas: Luis Carrero.
Suscripciones: (91) 250 15 93.
Antonio Zurdo.
Producción: Miguel Onieva.
Publicidad Madrid: Jerez, 3, Madrid-16.
María José Martín. (91) 457 45 66.
Nieves Fernández. (91) 457 45 66.
Publicidad Barcelona: Tallers, 62-64. Barcelona-1.
Enrique Alier. (93) 302 36 48.
Distribuye: SGEL. Avda. Valdelaparra, s/n. Alcobendas, Madrid.
Imprime: Novograph, S. A. Ctra. de Irún, Km. 12,450. Madrid.
Fotomecánica: Karmat. Pantoja, 10. Madrid.
Depósito Legal: **M-6622-1984.**

Año 1
Num. 2

Editorial

La revolución informática está aquí, llegando al ciudadano medio. Es imparable. Al igual que ocurrió con otras revoluciones, por ejemplo la del automóvil, nuevos modelos y usuarios ilusionados van incorporándose a su marcha imparable. La realidad avanza con pasos agigantados.

Durante este mes de abril, una importante cadena de grandes almacenes destinará mil metros cuadrados de una de sus sedes más concurridas a la exposición de la microinformática. Indudablemente, sabiendo prever el futuro que se aproxima, contribuye con su esfuerzo a implementar la informática en los hogares y pequeñas empresas españolas. Cuando menos, la mayor difusión del microordenador está asegurada.

Como es lógico, el incremento en la demanda trae consigo una mayor escasez de productos. Esto es lo que ha pasado con varios dispositivos de Commodore Business Machines. El imprevisible aumento de la demanda durante las pasadas navidades, unido a la importante disminución de la oferta de chips de RAM dinámica en los mercados internacionales, han llevado a **CBM** a producir sus propios chips de memoria en MOS Technology. Según parece desprenderse de esta nueva medida, la oferta de equipos, especialmente de modelos 64, está a punto de regularizarse. Por lo que sabemos, el distribuidor español se ha visto fuertemente desbordado. Esto dice mucho de nuestro mercado: empieza a perfilarse.

Lo que sí está claro es que Commodore está dispuesta a presentar batalla. Los modelos 264 y 364 parece ser que van a ser retrasados en su incorporación al mercado. Las fuentes bien informadas dicen que los nuevos modelos sufrirían la dura competencia del 64. Lo que sea, se verá.

En la Feria de Hannover, durante el mes de abril, se presentarán las novedades de la firma. Entre ellas se sitúan un sistema operativo parecido al UNIX, destinado al CBM-700, y un ordenador compatible con el PC de IBM. Añádase que es MOS Techonology, filial de CBM, quien fabricará sus propios microprocesadores 8088, bajo licencia de Intel. Seguiremos informando, dentro de unos días estaremos presentes en aquella importante feria alemana.

Claves para introducir en el ordenador los programas que aparecen en la revista.

COMO SE VE	COMO SE TECLEA	EFECTO CONSEGUIDO
	SHIFT + CLR	(LIMPIA PANTALLA)
	CLR	(HOME)
	CRSR	(CURSOR ABAJO)
	SHIFT + CRSR	(CURSOR ARRIBA)
	CRSR	(CURSOR DERECHA)
	SHIFT + CRSR	(CURSOR IZQUIERDA)
	CTRL + 1	(NEGRO)
	CTRL + 2	(BLANCO)
	CTRL + 3	(ROJO)
	CTRL + 4	(CIAN)
	CTRL + 5	(VIOLETA)
	CTRL + 6	(VERDE)
	CTRL + 7	(AZUL)
	CTRL + 8	(AMARILLO)
	CTRL + 9	(CARACTER INVERSO)
	CTRL + 0	(CARACTER NORMAL)

Centro COMMODORE
(SAKATI S.A.)

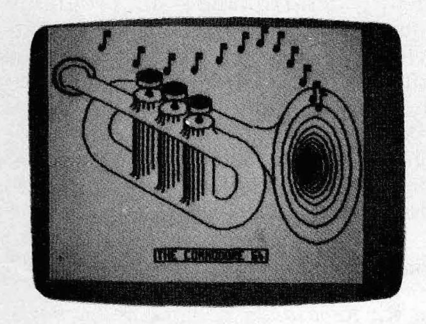
NOTICIAS DE SOFT

CONTROL DE ALMACEN - 64
Toda aquella empresa que tenga problemas de almacén no debe pararse en este anuncio. Debe llamarnos y le explicaremos con detalle este programa. El espacio no da para más, pero he aquí un avance:
— 1.200 referencias son controladas por el almacén, en unidades, litros, kilos, retros, etc., stock mínimo, y lo propio de estos programas, etiquetas, etc.
Pero además el programa tiene una opción de anotación para los gastos de su empresa, con análisis de rentabilidad de su negocio, sin pasar por contabilidad. De manejo sencillo. TODO EN CASTELLANO. El programa se presenta en disco.
P.V.P. 35.000 ptas.



EXTENSION DE BASIC - 64
Potencie el sistema operativo de su COMMODORE-64 con este cartucho que añade 42 nuevos comandos de gran utilidad. Trabaje con BASIC 4.0 de COMMODORE. Podrá localizar y corregir programas con gran facilidad, pasar programas de un CEM-800 a un 64, potenciar el editor, y conectar sin ningún interface, sólo con un cable, su 64 a una impresora CENTRONIC. MANUAL EN CASTELLANO.
P.V.P. 25.000 ptas.

LAPIZ OPTICO-64 Y VIC 20
Convierte su pantalla en una pizarra, podrá dibujar círculos, cuadrados, líneas, etc. MANUAL EN CASTELLANO.
P.V.P. 14.000 ptas. disco; 13.500 ptas. cinta.

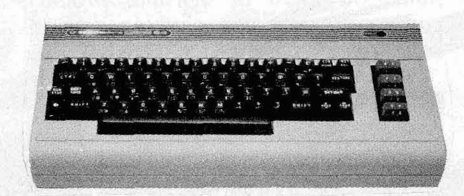


SUPERBASIC-64
Facilita el uso de pokes y peaks, dibuja en modo A.R. entrando líneas, círculos, etc. Define y manipula sprites, crea sonido o música, detecta colisiones, con sprite/sprite o sprite/fondo, lee directamente las puertas del joy stick paddle o lápiz óptico, dibuja usando un juego de gráficos turttle, imprime en una impresora commodore o epson compatible. Superbasic añade 50 nuevos comandos al basic. MANUAL EN CASTELLANO.
P.V.P. 7.500 ptas. disco; 6.900 ptas. cinta.

MAILING-64
Programa de correo, podrá crear hasta 1.200 nombres con direcciones, comentarios, informes, etc. Cree cualquier cantidad de ficheros maestros en discos separados y maneje cada uno de ellos con el mismo programa. Edite etiquetas con el criterio que usted defina, y búsqueda por orden alfabético. Número de registro, o claves de selección. MANUAL EN CASTELLANO.
P.V.P. 8.000 ptas. disco; 7.500 ptas. cinta.

JUEGOS
El Hortelano
Palillos
King Kong
Aterrizaje en Venus
Batalla Espacial
Comando Espacial
Caktus
Lago de los Tiburones
La Bruja
Guerra de Filones
Defensa Lunar

Cinta P.V.P. 2.000 ptas.
Ajedrez P.V.P. 4.000 ptas.



MONITOR CODIGO MAQUINA - 64
Programa realizado en código máquina que le permite ensamblar, desensamblar, crear puntos de ruptura, grabar en cassette o disco Incorpora el sistema operativo del disco (D.O.S.), compara bloques de memoria, enlace exterior, ejecuta un programa a partir de una dirección de memoria, busca en zona de memoria, visualiza el contenido de la memoria, etc. MANUAL EN CASTELLANO.
P.V.P. 6.000 ptas. cinta; 6.500 ptas. disco.

¿LE GUSTARIA COMPONER MUSICA?
Nuestro programa hará de usted un melómano. MUSICA -64 controla tres voces en un rango de 8 octavas cada voz. Sonarán en cualquiera de estos 4 instrumentos: órgano, clavicornio, silofón y trompeta. Podrá introducir 2.000 notas por voz, y una vez compuesta su música, archívela en cinta o en disco. INSTRUCCIONES EN CASTELLANO.
P.V.P. 4.000 ptas. en cinta; 5.000 ptas. en disco.

ARROW
Este programa hace posible la carga y el almacenamiento rápido de un programa en un cassette. Ahorrará así tiempo y espacio en cinta. Verifica si el programa se ha almacenado correctamente. Aporta un comando de posicionamiento para adelantar rápidamente la cinta. Calculador hexadecimal. Conversión instantánea a decimal.
Incluye ayudas de programación: incrementos para generar líneas de basic automáticamente, borrar líneas o grupos de líneas. Buscar una serie y listarla, reemplazar una serie por otra.
Monitor Código Máquina incorporado.
P.V.P.: 12.000 ptas. en cartucho.

SUPERBASE - 64
Es algo más que un potente generador de aplicaciones. Los programas por aplicación son limitados. Aporta un nuevo lenguaje con más de 40 nuevos comandos. Podrá diseñar su propia configuración, con registros de hasta 1.000 caracteres y gran facilidad de modificación de los mismos. 127 campos por registro. Contará con 15 ficheros por aplicación, siendo el número de programas por aplicación ilimitados. Los ficheros de clave y de memoria son así mismo ilimitados. Enlace con EASY SCRIP. Calendario aritmético. Esta Base de Datos es el número uno en su género, algo que usted debe tener para su trabajo. MANUAL EN CASTELLANO.
P.V.P. 22.500 ptas.

BUSICAL - 64
Es un potente programa de cálculo, similar al VISICALC, especialmente adecuado a cálculos financieros, escandallos, etc.
P.V.P. 8.000 ptas. en cinta.

BASE DE DATOS II - 64
Doce opciones presentadas en el menú. Cree nuevos ficheros de datos. Cuenta usted con 12 campos. Esta Base de Datos es ideal para personas que empiezan a introducirse en este método de trabajo.
P.V.P. 4.000 ptas. en cinta.

COMPACTOR
Es una rutina escrita totalmente en código máquina. Reducirá el tamaño de un programa BASIC, eliminando REM y caracteres redundantes y juntando afirmaciones formando líneas más largas. Desaparecen espacios, comillas o punto y coma innecesarios. No cambia la lógica del programa. El usuario puede obligar a la unión de dos líneas si así lo desea.
MANUAL EN CASTELLANO.
P.V.P.: 3.000 ptas. en cinta.

DESENSAMBLADOR - 64
Le permite desensamblar cualquier programa realizado en código máquina. INSTRUCCIONES EN CASTELLANO.
P.V.P. 2.000 ptas. en cinta.

BOLETIN DE PEDIDO

A enviar a SAKATI, S. A. - Ardemans, 24 - MADRID-28 - Tfno.: 256 77 94 - Telex: 44222.

Ref.		Cdad.	Precio
		TOTAL	

□ Talón adjunto □ Contra reembolso

Fecha: Firma:

Don ____________________

Calle ____________________ N.° ______

Código postal ______________ Ciudad ______________

Provincia ______________ Tfno: ______________

P.: En la actualidad me estoy diseñando un *plotter* de cara a conectarlo, o interfasarlo, con mi **Commodore 64.** Para mover el carro en las direcciones X e Y voy a utilizar motores paso a paso (carísimos por cierto) y un solenoide atrae al mecanismo que actua sobre el bolígrafo.

Mis preguntas son las siguientes: ¿Cómo puedo efectuar la conexión con el *port* del usuario? ¿Existe alguna contraindicación para que utilice la salida de 5 Voltios que tiene? ¿Es preciso que emplee optoaisladores para que el ordenador controle a los motores?

Ramón Fernández Alamo.

R.: *Comencemos con las fuentes de alimentación, que puede obtener a partir del* port *de usuario. La patilla número 2 le proporciona 5 voltios, pudiendo obtener hasta un máximo de 100 miliamperios. En las patillas 10 y 11 podrá obtener 9 voltios en cada una, siendo alterna la corriente en este caso, pero la máxima corriente es también de 100 mA. La diferencia estriba en la fase, que es opuesta entre ambas, tal como se conseguiría de un transformador/reductor.*

La utilización de optoacopladores puede resultar una buena idea, máxime porque se trata de un montaje experimental. De todas formas, es muy improbable que el tipo de motores paso a paso que utilice necesiten más de esos 100 mA, aunque podrían inducir espureos que alteren los datos manejados por el ordenador.

La utilización del port *del usuario no comporta graves dificultades. Entre las patillas C a L se sitúan las líneas de un bus de datos, que proceden del* port *B de la CIA, con las leyendas PB0 hasta PB7. La dirección de la memoria del* **64** *que nos da el acceso a ella es la 56577. Cada uno de esos 8 bits puede ser utilizado como entrada o salida de información, dependiendo si le corresponde un 1 o un 0 en el DDR, situado en la dirección de memoria 56579, en la cual un 1 significa salida y un 0 entrada.*

Sería conveniente que releyera el artículo que sobre el **Elektrocomputer** *publicamos en el número 1 de* **Commodore Magazine.** *Le aclarará varias ideas al respecto. A estos 8 bits, se le añaden 2 líneas de "apretón de manos", que darán su conformidad si la transmisión de datos ha sido correcta. La que resultará más interesante es la titulada FLAG 2, en la patilla B del* port, *que cuando se le aplique un flanco negativo de tensión, produce como respuesta una petición de interrupción al sistema.*

P.: Algunas veces he pasado programas que requerían sólo 3 K de expansión con el cartucho de 16 K conectado a mi **VIC-20**. Quisiera saber si existe alguna forma de pasar por alto este detalle.

Alvaro Muñoz Rivas.

R.: *Por el carácter de su pregunta se desprende que da por sentado que la configuración del mapa de memoria del* **VIC-20** *varía, según que cartucho de ampliación se conecte.*

El método para solventar el pequeño problema que le preocupa consiste en engañar al ordenador, haciéndolo creer que realmente tiene conectado el cartucho de 3 K. Para conseguirlo será cómodo recurrir a los clásicos POKE, alterando los contenidos de las direcciones adecuadas. No es necesario introducirlos en forma de programa, con su correspondiente número de sentencia, sino que bastará con teclearlos nada más conectar el ordenador.

P.: Quiero comprarme un **Commodore 64,** pero dentro de poco un familiar viajará a los Estados Unidos. He leído en algunas revistas que allí al precio es sustancialmente más bajo y un amigo me ha dicho que los sistemas de televisión son muy distintos en ambos países y por eso no me funcionará aquí. Si me aclaran la duda les quedaré agradecido.

Antonio Blanco.

R.: *Si usted se trae un equipo de fuera, sin pagar los correspondientes aranceles, estará cayendo en el contrabando. Aún suponiendo que el problema no existiera, debemos decirle que su amigo lleva razón. Varias son las diferencias entre los estándares de televisión adoptados en ambos lados del Atlántico.*

Mientras que en España se utiliza el sistema **PAL** (Phase Alternating Line) *para el color, en los Estados Unidos se utiliza el* **NTSC,** *que trabaja de una manera fundamentalmente distinta.*

Por otro lado, le sonará el concepto de 625 líneas empleado aquí. Esto quiere decir que la imagen en la pantalla queda formada por 625 líneas paralelas, que varían en intensidad para generar la imagen completa. En aquel país solamente se utilizan 525 líneas, por lo que tampoco hay compatibilidad en los sincronismos para generación de las imágenes. Así que lo único que conseguirá son extrañas imágenes en la pantalla.

P.: Tengo un ordenador **VIC-20**. Estoy desarrollando un programa que utiliza varias sentencias que incluyen INPUT, para introducir datos durante la ejecución. Como es lógico aparece el signo de la interrogación. Por la presentación que lleva el programa, resulta incómodo que haya tantas. ¿Podrían indicarme algún método para eliminarlas?

Santiago Arroyo.

R.: *Repasando el mapa de memoria del* **VIC-20** *encontramos que la dirección 19 ($0013) actua como flag que indica cuál es el símbolo que se utilizará como prompt* (apuntador). En otras palabras aquí lee el ordenador qué símbolo aparecerá en la pantalla, para "apuntar" donde se situa el cursor en ese instante.

Si el contenido de esta dirección es O, entonces aparece la ?. En otro caso desaparece. A grandes líneas, la solución a su problema consiste en alterar este contenido. Por otro lado, si variamos la información mediante POKE, aparecerían efectos colaterales, por tanto, hay que volver a ponerlo a O, una vez conseguido el efecto deseado.

También es recomendable limpiar el buffer *del teclado, para evitar alteraciones debidas a falsos tecleos efectuados antes del INPUT. La dirección 198 ($0006) almacena el número de caracteres que hay en el* buffer, *poniéndola a O solventa el tema (POKE 198,0).*

La subrutina a utilizar sería del siguiente tipo:

P.: Recientemente he comprado un cassette para mi **64**, conocido como **C2N**. Creo que es un modelo nuevo. Como me indicó el vendedor, la conexión se realiza por medio de un conector único. Sin embargo, al preguntar que dónde encja el cable plateado que sale del enchufe, no supe darme una respuesta. El cassette funcionó perfectamente en la tienda y también lo sigue haciendo en mi casa, pero el cable "colgante" sigue siendo un misterio, ya que en las instrucciones no se hace mención de él.

Nicolás García Navarro.

R.: *Si el aparato trabaja perfectamente no se preocupe. En realidad, se trata de un hilo en forma de malla, destinado a una toma de masa. Por el se derivan las corrientes eléctricas parásitas, que dificultan la carga de programas en el ordenador. Si no tiene esos problemas con la electricidad estática, déjelo en paz. Su misión es muy similar a la de las cadenitas y correas que se ponen bajo algunos automóviles, para derivar las corrientes a tierra.*

P.: Normalmente he leído que el microprocesador **6502** puede direccionar hasta 64.000 posiciones de memoria. El **VIC-20**, que poseo, creo que lo lleva en su interior, pero **Commodore** sólo dispone de ampliaciones de 16 K máximo. ¿Cuál es la máxima cantidad de memoria disponible?

Javier Delgado Nieto.

R.: *La máxima memoria accesible por el usuario es de 32 Kbytes en el* **VIC**, *como consecuencia de su diseño. Sin embargo, no toda esta cantidad está siempre a su disposición. Por ejemplo, el intérprete del* **BASIC** *ocupa parte de esta cantidad (unos 3 Kbytes), con lo cual sólo están disponibles 29 K a los que poder acceder. Siempre y cuando se cuente con los cartuchos de ampliación de RAM adecuados. Los cartuchos de programas también ocupan parte de estas direcciones.*

HES MON 64

Monitor de depuración y control

Programar en Lenguaje de Máquina es siempre una tarea arduä, pero en ocasiones, inevitable. Allí donde el BASIC no se muestra suficientemente rápido o preciso, no queda más remedio que acudir al repertorio de instrucciones del *6510* para extraer de él todo su potencial, tanto en velocidad como en reducción de tamaño. Situaciones como tratamiento de interrupciones, manejo de periféricos, optimización de recursos de la máquina, etc. aconsejan abandonar la comodidad e interacción del lenguaje de alto nivel interpretado, por el áspero pero extremadamente efectivo lenguaje de máquina.

Bien es sabido que desde el BASIC es posible cargar en memoria y después ejecutar, subrutinas en lenguaje de máquina. Basta para ello utilizar instrucciones POKE que vayan situando en direcciones de memoria consecutivas cada uno de los bytes que componen nuestro programa en lenguaje de máquina. Pero, claro está, el BASIC ignora la naturaleza y estructura interna de este programa y se limita a hacer lo mismo que haría con cualquier otro tipo de tabla que quisiéramos ubicar en memoria, a partir de una cierta dirección.

Esto quiere decir que somos nosotros quienes, tras escribir el programa, en lenguaje de máquina, hemos de actuar de "ensambladores", poniendo primero las cosas tal como deben quedar en memoria; para una correcta ejecución colocar primero el byte de menor peso de las direcciones absolutas y luego el de mayor, sustituir los saltos a otras instrucciones dentro de la rutina por desplazamientos relativos, traducir los códigos nemotécnicos de operación a sus valores hexadecimales según el tipo de direccionamiento utilizado, etc. Y a continuación, claro está, pasar cada uno de estos valores hexadecimales a decimal, para montar la correspondiente instrucción DATA.

El sistema, aparte de laborioso e incómodo, es sumamente propenso a que se deslicen errores. Por eso, cuando se trabaja regularmente con programas en lenguaje de máquina de cierto tamaño no se puede pasar mucho tiempo sin dos herramientas fundamentales: un programador ensamblador y un monitor de depuración y control. *HES MON 64* pertenece a la segunda categoría y puede considerarse, además, como un ensamblador "mínimo".

Una sesión de trabajo con HES MON 64

Un pequeño ejercicio en lenguaje de máquina nos permitirá ir examinando los comandos con que cuenta *HES MON 64.* Se trata de leer el contenido de los tres bytes que componen el reloj de tiempo real del sistema, y presentarlos en pantalla en una posición fija (ni siquiera pretendemos mostrar un "auténtico" reloj de horas, minutos y segundos, para no alargar el ejemplo).

Al conectar el *Commodore 64*, habiendo colocado previamente el cartucho de *HEX MON 64* en el *slot* de ampliación, lo primero que aparece en pantalla es lo siguiente:

```
C*
PC IRQ SR AC XR YR SP
0000 EA31 27 00 00 FA
```

La C es indicativa de un "arranque en frío" del programa y en las dos líneas que siguen aparece el contenido de ls registros del procesador (PC: contador de programa, SR: registro de estado, AC: acumulador, XR: registro X, YR: registro Y, SP: puntero de la pila), además del puntero IRQ que contiene la dirección de la rutina de atención a las interrupciones.

Para empezar a meter instrucciones a partir, por ejemplo, de la dirección 1200 (hex.) utilizamos la función *A* (de Assemble, es decir, ensamblar una instrucción de máquina):

```
A1200 JSR FFDE
```

y al pulsar Return nos aparecería en pantalla:

```
1200 20 DE FF JSR $FFDE
1203
```

esto es, la instrucción ensamblada (20: código de operación, DE: byte bajo, FF: byte alto), seguida de la instrucción en formato nemotécnico, tal como la introdujimos. Además, el contador de programa se nos queda preparado en la primera posición libre (1203) a la espera de que demos

la siguiente instrucción, ya sin necesidad de precederla de la orden *A*.

De este modo proseguiríaos metiendo instrucciones hasta el final de nuestro programa. Para visualizar y comprobar lo introducido utilizamos el comando *D* (Disassemble o desensamblar), precisando entre qué direcciones queremos hacerlo. El listado que obtendríamos sería el de la figura 1.

Como vemos, en este listado se llama a tres subrutinas del *KERNAL* del *CBM64* ($FFDE que lee el reloj del sistema; $FFD2 que saca a pantalla el contenido del acumulador, y $FF0 que sitúa el cursor en un punto X-Y de la pantalla) y a otra cuya dirección es $1240, que utiliza a su vez una tercera que empieza en $1270. Escribimos a continuación estas dos últims rutinas (utilizando de nuevo *A*). (Ver figura 2 y 3).

La misión de estas dos rutinas es convertir a dos caracteres hexadecimales en ASCII el contenido del acumulador y presentarlos en pantalla llamando a $FFD2. El primer segmento, por su parte, se ocupa de enviar sucesivamente a la rutina de conversión y presentación, el byte de mayor orden del reloj (que FFDE deja en el registro Y), el de orden intermedio (en el registro X) y el de menor orden (en el acumulador), además de fijar las coordenadas de la pantalla donde aparecerán estos valores.

Tres modalidades de ejecución

Para ejecutar nuestro programa-ejemplo, *HES MON 64* nos ofrece tres modalidades: sin paradas (comando *G*), instrucción a instrucción (comando *W*), y con posibilidad de establecer un *"breakpoint"* en una instrucción cualquiera del programa (*Q* combinado con *B*).

La ejecución paso a paso (*W*) nos permite hacer un seguimiento o *"trace"* completo de nuestro programa. Por cada instrucción ejecutada el monitor nos presenta dos líneas: una con el contenido de los registros (R, de Estado Acumulador; X, Y, Puntero de la pila), y otra con la instrucción que se ejecutaría a continuación y la dirección de memoria (contador de programa) donde está. La ejecución del programa queda detenida a la espera de que se pulse una tecla para continuar. Cuando se entra en una subrutina que no interesa recorrer instrucción a instrucción, pulsando *J* se avanza hasta la instrucción RTS. Este es el aspecto que ofrece el seguimiento de la ejecución de nuestro programa (observe que para pasar a STX 1290 sin visualizar toda la subrutina FFDE, hemos tenido que pulsar J al entrar en ésta). Ver la figura 4.

Esta modalidad puede resultar muy interesante para observar el comportamiento del programa en los saltos condicionales, la evolución de los registros utilizados como contadores, las condiciones en que las variables del programa entran y salen de las subrutinas, etc. Como, tras la ejecución de cada instrucción, el programa cede control al monitor, nada nos impide entonces modificar el contenido de los registros o de ciertas posiciones de memoria, al objeto de verificar la marcha del programa en las situaciones límite que deseemos.

Los registros se modifican sin más que escribir ";" y a continuación los nuevos contenidos, en el orden en que el monitor los presente al pulsar la orden *R*. En cuanto a la memoria, su contenido se visualiza con *M* (dirección de comienzo) (dirección de final), mientras que su modificación se hace con el comando ";" seguido de la dirección a partir de la cual se va a hacer y de los valores a introducir.

En nuestro caso, por ejemplo, observamos que hemos elegido unas coordenadas de presentación del reloj inadecuadas, y las cambiamos modificando el segundo byte de las instrucciones LDX y LDY correspondientes:

: 120D 14
: 120F OF

```
D1200 1230
,1200 20 DE FF JSR $FFDE
,1203 8E 90 12 STX $1290
,1206 8C 91 12 STY $1291
,1209 8D 92 12 STA $1292
,120C A2 21    LDX #$21
,120E A0 0C    LDY #$0C
,1210 18       CLC
,1211 20 F0 FF JSR $FFF0
,1214 AD 91 12 LDA $1291
,1217 20 40 12 JSR $1240
,121A A9 58    LDA #$58
,121C 20 D2 FF JSR $FFD2
,121F AD 90 12 LDA $1290
,1222 20 40 12 JSR $1240
,1225 A9 58    LDA #$58
,1227 20 D2 FF JSR $FFD2
,122A AD 92 12 LDA $1292
,122D 20 40 12 JSR $1240
,1230 4C 00 12 JMP $1200
,1233
```

```
D1240 1254
,1240 48       PHA
,1241 4A       LSR
,1242 4A       LSR
,1243 4A       LSR
,1244 4A       LSR
,1245 20 70 12 JSR $1270
,1248 20 D2 FF JSR $FFD2
,124B 68       PLA
,124C 29 0F    AND #$0F
,124E 20 70 12 JSR $1270
,1251 20 D2 FF JSR $FFD2
,1254 60       RTS
,1255
```

Otro comando interesante para modificaciones es *H*, que se utiliza para buscar una determinada secuencia dentro de un intervalo de direcciones. Tiene dos modalidades:

H 1111 2222 33 44 55...
H 1111 2222 "ABCDE..."

La primera busca la secuencia de caracteres cuyos códigos ASCII son 33 44 55 entre las direcciones 1111 y 2222, de la memoria. En la segunda, los caracteres que hay que buscar no se expresan por su código, sino directamente encerrados entre comillas. En nuestro caso, hemos empleado *H*

HES MON

para localizar la posición de los caracteres X ($58) que empleamos como separadores de los tres bytes del reloj para sustituirlos a continuación por ";" ($3A). *H* nos devuelve todas las direcciones donde aparece el carácter o la secuencia que buscamos, evitándonos de este modo tener que hacer un recorrido exhaustivo por todo el programa.

Cuando nuestro interés en el comportamiento del programa se centra en algún punto muy concreto *HES MON 64* nos permite situar un punto de parada (*"breakpoint"*) en una instrucción determinada, de manera que el programa se ejecute normalmente hasta llegar a dicho punto.

El punto de parada se establece con el comando *B:*

B 120C 0001

sitúa un punto de parada en la instrucción 120C. El otro parámetro indica el número de veces que se ejecutará la instrucción antes de que el monitor detenga la ejecución al pasar por ese punto (en este caso, una vez).

Estos puntos de ruptura sólo son reconocidos por el monitor cuando la modalidad de ejecición es *Q ("Quick Trace")*. *Q* puede ir acompañado de la dirección donde se desea que comience el trazado; si no se da ninguna, comienza en la posición actual del contador de programa. Cuando se alcanza el punto de parada establecido la ejecición del programa entra automáticamente en la modalidad *W* (paso a paso, con presentación de registros y de la instrucción siguiente).

Cambio de ambiente: BASIC - Lenguaje de Máquina y viceversa

HES MON 64 ofrece la posibilidad de trabajar simultáneamente en código de máquina y en BASIC, pasando de una manera cómoda de uno a otro. El comando *X* cede control al intérprete de BASIC, sin pérdida de los programas ensamblados. Una vez en BASIC, pulsando la tecla RESTORE, el control vuelve a *HES MON 64*, y podemos seguir trabajando en lenguaje de máquina.

Para probar estas facilidades pasamos a BASIC y escribimos este pequeño programa con la intención de ver en decimal nuestro "reloj hexadecimal". Ver figura 5.

En el cálculo de los equivalentes decimales de la dirección de comienzo de la subrutina ($1200=4608) y de las direcciones de memoria donde guardamos los registros a la vuelta de FFDE ($1290 y $1292), *HES MON 64* nos facilaita la tarea, con sus funciones de conversión de hexadecimal a decimal ($) y de decimal a hexadecimal (#).

Al ejecutar este programa advertimos de inmediato que la lectura del reloj no aparece en decimal, como quizá pudiéramos esperar tras un examen poco atento. La razón, lógicamente, es que la subrutina llamada no regresa al programa BASIC, ya que está construida de manera que se repita continuamente (JMP $1200 en la dirección $1230). Como, además, incorpora sus propias instrucciones de salida, los datos siguen apareciendo en hexadecimal. Para que las cosas resulten como queremos, salimos del BASIC pulsando RESTORE. De nuevo bajo control del monitor vamos a modificar nuestra rutina en l. de m. De hecho, las únicas instrucciones que ahora necesitamos son la llamada a la rutina de lectura del reloj de tiempo real ($FFDE) y las de almacenamiento de los registros en memoria, a las que el BASIC accederá mediante PEEKs. Así pues, en la dirección $120C introducimos (mediante el comando *A*) una instrucción RTS, de manera que cada llamada SYS 4608 del programa BASIC ejecutará la rutina hasta este punto, ignorando todo lo demás (conversión y dalida, que ahora es realizado automáticamente por el BASIC).

El comando *XC* nos pasa de nuevo a BASIC. Volvemos a ejecutar de nuevo y ahora las lecturas del reloj aparecerán ya en decimal.

```
D1270 1279
,1270 C9 0A      CMP #$0A
,1272 90 03      BCC $1277
,1274 69 36      ADC #$36
,1276 60         RTS
,1277 09 30      ORA #$30
,1279 60         RTS
,127A
```

```
;1200
W

 20 06 14 0F A8
,FFDE 4C DD F6 JMP $F6DD
 20 63 2D 06 AA
,1203 8E 90 12 STX $1290
 20 63 2D 06 AA
,1206 8C 91 12 STY $1291
 20 63 2D 06 AA
,1209 8D 92 12 STA $1292
 20 63 2D 06 AA
,120C A2 14    LDX #$14
 20 63 14 06 AA
,120E A0 0F    LDY #$0F
 20 63 14 0F AA
,1210 18       CLC
 20 63 14 0F AA
,1211 20 F0 FF JSR $FFF0
```

```
5 SYS 4608
10 PRINTPEEK(4752);":";PEEK(4753);":";PEEK(4754)
15 PRINT"TT"
20 GOTO 5

READY.
```

Otros comandos

Estos son algunos de los comandos adicionales de *HES MON 64* que no hemos citado hasta ahora:

— *C* (comparar bloques de memoria).
C 1111 2222 3333

Compara el bloque de memoria comprendido entre 1111 y 2222 con el del mismo tamaño que comienza en 3333. Muy útil, por ejemplo, para comparar las diferencias entre dos versiones del mismo programa en lenguaje de máquina.

— *F* (rellenar un bloque de memoria).
F 1111 2222 33

Asigna el valor 33 a todas las posiciones del bloque de memoria comprendido entre 1111 y 2222.

— *I* (interpreta memoria).
I 1111 2222
Presenta los contenidos de memoria de las direcciones 1111 a 2222 en forma de caracteres ASCII, a razón de 32 por línea.
— *L* (cargar en memoria un programa).
L "programa" 11

Carga del dispositivo externo 11 (cinta o disco) un segmento de código (que puede ser o no un programa).

— *S* (guarda en un dispositivo externo una zona de memoria).
S "fichero" 11 2222 3333

La zona de memoria comprendida entre las direcciones 1111 y 2222 es guardada con el nombre "fichero" en el dispositivo externo 11 (01 para la cinta y 08 para la unidad de disquete).
— *T* (transfiere un bloque de memoria).
T 1111 2222 3333
Transfiere el bloque limitado por las dos primeras direcciones a la zona que comienza en 3333.
— *N* (ajuste de direcciones).
N 1300 1380 0100 1280
Supongamos que hemos copiado (con *T*) un programa que teníamos en 1200-1280, a la zona 1300-1380. En la copia las referencias a direcciones del interior del programa estarán afectadas por esa diferencia de $0100 en los orígenes de ambas zonas. Lo que hace el comando *N*, tal como lo hemos escrito en ejemplo, es añadir el valor $100 a cualquier referencia a direcciones entre 1200 y 1280, que encuentre en el programa que hay en la zona 1300-1380.
— *P* (copia a impresora el contenido de la pantalla).

Gumersindo García

POKEs MUY UTILES

● Si no quieres que nadie interrumpa tus programas en ejecución, antes de ponerlo en marcha efectúa el siguiente POKE:
POKE808, 225, que inutiliza la secuencia RUN/STOP-RESTORE. Para volverla a habilitar introduce:
POKE 808, 235

● Si no te gusta que se autorrepitan algunas teclas, puedes evitarlo con:
POKE 650, 64, no se repetirá ninguna.

Si, al contrario, lo que deseas es tener todas las teclas autorrepetitivas ejecuta:
POKE 650,128; y para dejarlo toto otra vez como estaba (Cursor, barra espaciadora e INST/DEL, repetitivas) no tienes más que efectuar:
POKE 650,0

● Con:
POKE 204,0 antes de una instrucción GET, se producirá el efecto del cursor parpadeando, esperando respuesta. Para "apagar" el cursor usa:
POKE 204,1

También puedes hacer desaparecer el cursor parpadeante que sale al efectuar una instrucción INPUT. La forma de conseguirlo es efectuando: POKE 207,1, delante de la instrucción mencionada. (Con POKE 207,0 dejarás al cursor como está).

¿CUAL ES EL FORMATO DEL DISCO?

Las unidades de diskette **4040** y **1541** no son compatibles entre sí. Un *diskette* formateado en el primero, sufre cuando posteriormente se le utiliza con el **5141**. Esto queda detectado por indicaciones extrañas producidas por el LED y los chirrientes sonidos que produce el motor paso a paso, de posicionamiento de la cabeza. Los mensajes de DISK ID MISMATCH son claros, hasta que el diskette termina por convertirse en inútil. Si la operación se realizase a la inversa, los resultados resultarán similares.

La tragedia reside en que, a simple vista, los *diskettes* son incapaces de indicar en qué tipo de unidad fueron escritos y formateados. La razón fundamental para la incompatibilidad reside en que las marcas de sincronismo, que se incluyen en cada sector del *diskette,* tienen naturaleza diferente. Así, el **4040** formatea de otro modo que el **1541.**

El **4040** llena todo el *diskette* con ceros. Por el contrario, el **1541** lo llena de unos y pone los punteros de pista siguientes a 75. De aquí se explica por qué los distintos mensajes del *diskette* incluyen algo como ILLEGAL TRACK AND SECTOR 75, y el directorio en principio está repleto de ficheros bautizados como AAAA, donde cada uno consta de 256 bloques.

Partiendo de la base de que el *diskette* no está lleno, nos basaremos en esta información, para determinar en qué tipo de unidad de disco ha sido formateado.

Utilizando el programa que aparece acompañando al texto, el **1541**m proporcionará lo siguiente: 75 1111111. Por el contrario, el **4040** facilita 8 ceros.

Podría ocurrir que la respuesta fuese otra cosa diferente a una serie de ceros o unos. En tal caso se debe a que no hemos leído en un sector vacío. La solución consiste en cambiar de pista (P) y/o sector (S) en el programa, hasta dar con uno vacío. La solución consiste en cambiar de pista (P) y/o sector (S) en el programa, hasta dar con uno vacío. Podría ocurrir que por más vueltas que se le dé no lleguemos al resultado apetecido. En tal caso habrá que empezar a dudar de la procedencia del diskette y su contenido. Este programa puede evitar serios disgustos.

```
10 A$=CHR$(0)
20 REM***PERIFERICO,DRIVE
30 PE=8:D=0
40 REM***PISTA,SECTPR
45 PRINT"□"
50 INPUT"PISTA ";P:INPUT"SECTOR ";S
60 OPEN 15,PE,15:OPEN1,PE,3,"#"
70 GOSUB140:IFETHENSTOP
80 PRINT#15,"U1"3;D;P;S
90 GOSUB 140 :IFETHENSTOP
100 FORJ=0TO7:GET#1,I$
110 PRINT ASC(I$+A$)
120 NEXTJ
130 CLOSE1:CLOSE15:END
140 INPUT#15,E,E$
150 PRINTE;E$
160 RETURN
```

PROGRAMA PARA DETECTAR FORMATO DE DISKETTES

COPIA SIN MIEDO

Algunos programas de los que incluimos en la revista pueden resultar tediosos de teclear, lo cual se agrava más todavía si tenemos la denominada "fobia" a los programas de revista. Pues bien, todo puede resultar cuestión de paciencia porque lo que sí aseguramos es que los programas publicados funcionan perfectamente.

Es más, los listados presentados son los que proporciona la impresora, después de haberse ejecutado el programa de modo satisfactorio, por lo cual NO se dan lugar las erratas de imprenta, puesto que no pasan por ella.

Aparte de todo, ni el más experto programador del mundo sería capaz de asegurar después de copiar un programa en un ordenador, que este vaya a funcionar: "Errar es humano".

Así que copia sin miedo, guarda el programa en tu *cassette* o *diskette*y después pruébalo; si no funciona, repasa, seguro que se te ha escapado algo. Y aunque muchas cosas se te

habrán ocurrido ya para copar, te diremos algunas que te podrán ser de utilidad:

— Asegúrate de que sabes escribir todos los símbolos del listado. Para ello puedes ayudarte de las claves que siempre incluimos al principio de cada número.

— Aprovecha la facilidad de tu **CBM** para copiar líneas, cuanto te encuentres con varias parecidas.

— ¡Ojo con las comillas! Cuando vayas a insertar algo dentro de un texto entrecomillado, no olvides que no dispones del control del movimiento del cursor una vez pulsado INST/DEL. Consulta con tu manual, y por si no lo sabías, si te lías dentro de las comillas, pulsa SHIFT/RETURN, la línea no se almacenará, quedando como estaba antes de estropearla y tendrás de nuevo control sobre el cursor.

— Si te ayudas de una hoja para ir destapando línea a línea, según las vayas copiando mejor. Con ello seguro que evitarás el confundir una línea con otra..

— Si el programa que vas a copiar contiene muchas instrucciones DATA, una forma rápida y segura, y que al final te servirá para repasar más cómodamente, es grabarlas con tu propia voz en cinta todas seguidas; luego con escuchar y teclear basta.

Esperamos que estos consejos te sean de utilidad.

ASI SE CONGELA LA PANTALLA

Cuando se lista un programa en el **64**, se puede hacer que la visualización vaya más despacio, simplemente presionando la tecla CTRL.

Con este breve programa se puede congelar el contenido de pantalla, con solo presionar las teclas SHIFT o SHIFT LOCK.

Corra el programa, después teclee NEW (RETURN). Cargue el programa, y escriba LIST (RETURN). A partir de ahora, cada vez que presione SHIFT, el listado se detendrá.

```
10 REM***RUTINA DE PAUSA
20 FOR I=49152 TO 49161:READ A:POKE I,A:NEXT
30 DATA 72,173,141,2,208,251,104,76,26,167
40 POKE 774,0:POKE 775,192:END
```

UN VIC CON ACELERADOR VIC-20

Esta pequeña rutina sirve como acelerador, es decir, permite modificar a voluntad la velocidad de ejecución de un programa. El corazón de la rutina es la instrucción POKE 37879,K donde K puede tomar cualquier valor entre cero y 255. Según el valor que se introduzca en la posición 37879 de memoria, la velocidad aumenta o disminuye.

Esta modificación se puede hacer tanto como comando directo, como desde dentro de un programa, lo que puede ser útil en programas de juegos para que, una vez que se alcanza una puntuación determinada, el programa empiece a funcionar mucho más deprisa, o más despacio. Claro que toda mejora siempre presenta alguna desventaja más o menos importante. En este caso al modificar la velocidad del sistema el reloj de tiempo real también modifica su velocidad y deja de ser de tiempo real. Así que hay que decidir entre una modificación en velocidad y el uso del reloj en tiempo real.

```
0 REM***UN VIC CON ACELERADOR***
10 PRINT"J":CLR:PRINT"00-32    MUY DESPACIO"
20 PRINT"33-65    DESPACIO"
30 PRINT"66-128   MEDIO"
40 PRINT"129-190 RAPIDO"
50 PRINT"191-255 MUY RAPIDO"
60 PRINT:PRINT:PRINT
70 INPUT"COMO DE RAPIDO";K
80 IFK>0 ANDK<256 THEN 100
100 POKE37879,K
150 REM***RUTINA PARA DEMOSTRACION***
200 PRINT"J":FOR J=1 TO 100:PRINT J:NEXT
300 GOTO 10
```

PRINT AT X, Y 64

Esta es una instrucción que echamos de menos los usuarios del **CBM-64.** Con ella se puede imprimir directamente en cualquier posición de la pantalla, con sólo indicarle las coordenadas del punto a partir del cual queremos que empiece a escribir. Pues bien, hay varias formas de simularla y aquí te incluimos dos de ellas en BASIC, a ver si se te ocurre alguna más...

La primera de ellas es:

10 POKE 214, Y:POKE 211, X-1: PRINT CHR$ (145); Z$, o bien
10 POKE 214, Y: PRINT CHR$ (145); TAB(X-1); Z$

La otra manera:

10 Y$="(25 veces cursor-abajo): X$="(40 veces cursor-derecha)
20 PRINT"(cursor-HOME)"; left$ (Y$,Y); LEFT$(X$,X); Z$

Siendo, en ambas, X el valor de la coordenada horizontal e Y el de la coordenada vertical (ten en cuenta que nuestra pantalla es de 40 × 25); y Z$ la cadena de caracteres que se quiere imprimir.

Para probar cualquiera de las dos formas teclea:

5 INPUT "X"; X:INPUT "Y"; Y: INPUT "Z$"; Z$; PRINT "(CLR/HOME)"

y a continuación las líneas arriba sugeridas.



Cuando se tiene 64 K de memoria, una magnífica resolución, 16 colores, efectos tridimensionales con sprites, un sonido equivalente al de un sintetizador, un teclado profesional con 62 caracteres gráficos, toda una amplia gama de periféricos, la más completa gama de programas educativos, profesionales y de video-juegos...; en resumen, cuando se es un ordenador personal como no existe ningún otro en el mercado y el más vendido mundialmente, es muy difícil decir sin orgullo que eres un Commodore-64.

Claro que más difícil todavía es decir sin orgullo que tienes un Commodore-64. ¿Por qué no lo comprueba?

COMMODORE 64 LE DA ACCESO A MUCHOS ACCESORIOS

Unidad simple de disco (Monofloppy) 170 K.

Cassette.

Plotter e impresora, 4 colores, 14 c.p.s.

Impresora matricial, tractor, 30 c.p.s.

Interface RS232.

Joy Stick.

Paddle.

Cursos de Introducción al BASIC.



COMMODORE 64 LE MUESTRA PARTE DE SUS PROGRAMAS

Utilitarios y lenguajes

MONITOR LENGUAJE MAQUINA.
FORTH.
LOGO.
PILOT.
MACRO ASSEMBLER.
PROGRAMMER'S UTILITIES.
TURTLE GRAPHICS II.
MASTER.

Sistemas operativos

FILE/BOSS.
CP/M.

Programas de aplicaciones

EASY SCRIPT.
Proceso de texto de gran potencia.

CALC RESULT.
Hoja electrónica de cálculo.

EASY CALC RESULT.
Versión simplificada del CALC RESULT.

MAGIC DESK.
Proceso de texto y gestión de ficheros.

AGENDA TELEFONICA.

Programas educativos

MUSIC MACHINE.
MUSIC COMPOSER.
VISIBLE SOLAR SYSTEM.
SPEED/BINGO MATH.
FISICA I.
MATEMATICAS I.
HISTORIA I.
GEOGRAFIA I.
GEOGRAFIA II.
JUEGOS EDUCATIVOS.
TEMAS MONOGRAFICOS.
CONOCIMIENTOS GENERALES.
QUIMICA I.

Juegos

JUPITER LANDER.
KICKMAN.
SEAWOLF.
RADAR RAT RACE.
TOOTH INVADERS.
LAZARIAN.
OMEGA RACE.
LE MANS.
PINBALL SPECTACULAR.
AVENGER.
SUPERMASH.
FROGMASTER.
GRID RUNNER.
ATTACK OF THE MUTANT CAMELS.
THE PIT.
MR. TNT.
6 GAME PROGRAMS.
BINGO.
ROOTING TOOTING.
MINESSOTA FAT'S POOL CHALLENGE.

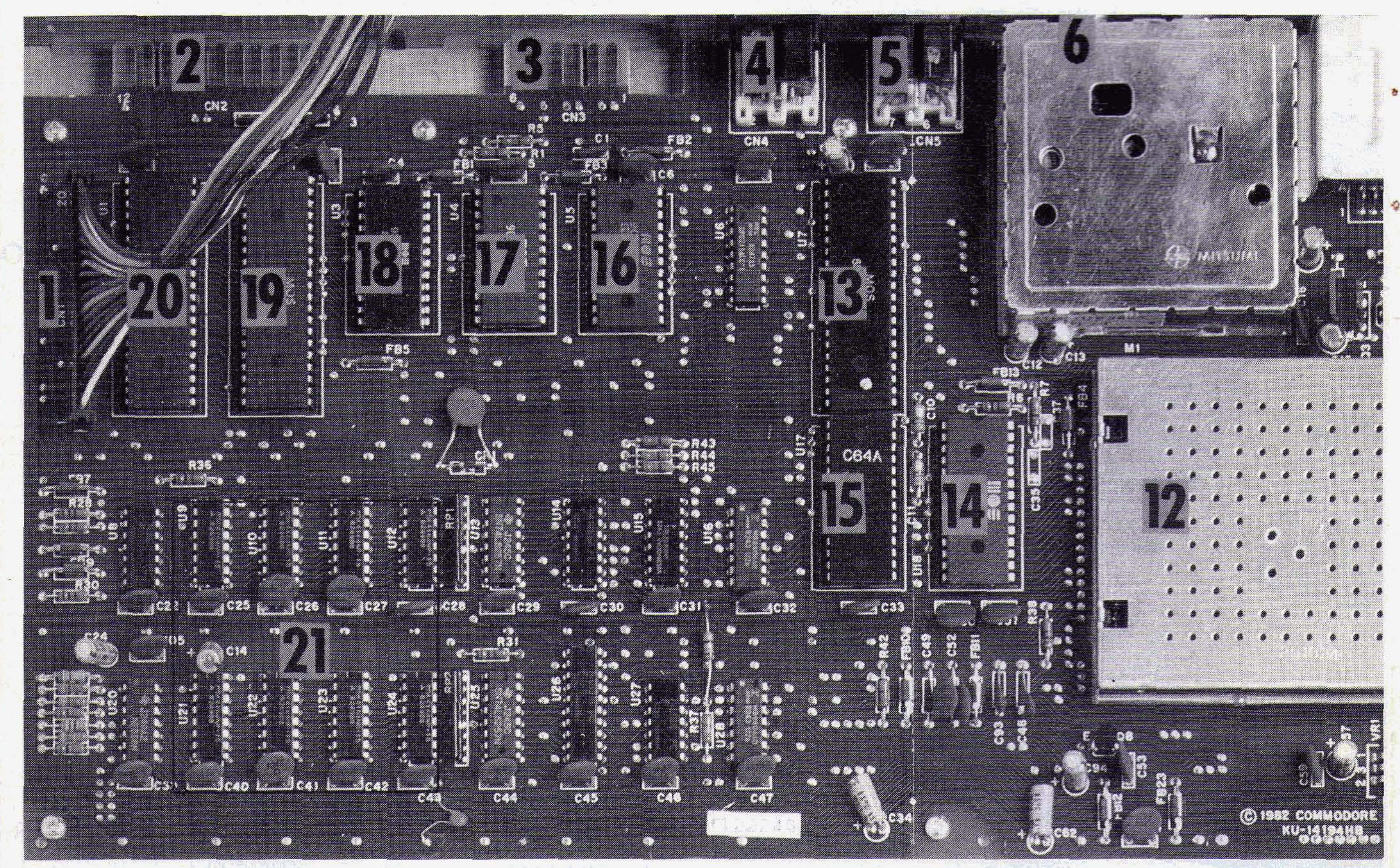
... y seguimos ampliando la lista

El ordenador personal de la familia más potente



MICROELECTRONICA Y CONTROL, S.A.
c/. Taquígrafo Serra, 7, 5.°. Barcelona-29
c./ Princesa, 47, 3.° G. Madrid-8

CBM 64 en profundi



1. Este es el conector del teclado, que posibilita que el mismo pueda ser leído por el ordenador.

2. Port del usuario. Se trata de un conector de 24 patillas, capaz de acceder a una amplia gama de dispositivos.

3. Port del cassette. Es un *port* con 12 patillas, que sólo acepta el conector estándar desarrollado por **Commodore** para su unidad de *cassette.* Sin embargo unas simples modificaciones permitirían la utilización de otras unidades no especiales, suponiendo que se disponga de un *interface* adecuado.

4. Port serie RS232. Es un *port* de 6 patillas, utilizado para la comunicación con la unidad de *diskettes* o la impresora. La existencia de *ports* similares en esos periféricos, permiten la conexión en cadena.

5. Conector de audio/video. Enlaza al **64** con una serie de dispositivos, incluyendo un equipo de alta fidelidad, utilizado cuando quiera escuchar mejor el *chip* SID.

6. Conector para el televisor.

7. Slot para cartuchos. Es un conector de dos hileras de 22 patillas, utilizado principalmente para conectar cartuchos de juegos, aunque puede ser empleado para conectar diversos *interfaces.*

8. Zócalo para la alimentación. Une al **64** con su transformador, y así con la alimentación.

9. Interruptor de encendido y apagado.

10. Port de juegos número 2. Es un *port* de 9 patillas, que permite que puedan ser conectados *joysticks, paddles,* lápices ópticos, etc.

11. Port de juegos número 1. Como el anterior, pero codificado por una parte diferente del intérprete del BASIC.

12. El 6566. Se halla cubierto por una unidad disipadora de calor. Este VIC *(Video Interface Chip)* es similar al *chip* VIC del modelo **VIC-20** sólo por su nombre, aquí es un dispositivo mucho más potente.

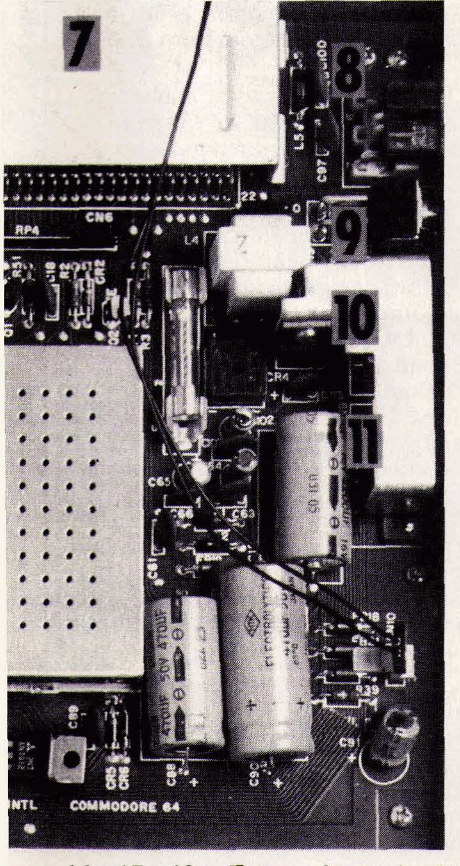
13. El 6510. Es el procesador principal que controla al **Commodore 64,** manejando gran cantidad de elementos que entran y salen de él. En el *chip* se halla un *port* de Entrada/Salida bidireccional de 8 bits, siendo programable bit a bit.

La arquitectura interna del procesador es idéntica al del **6502** anterior de **MOS Technology,** aunque se han cambiado los primeros 2 bytes. Estos no son de RAM, sino que permiten el control de algunas de las actividades del **64.** Por ejemplo, detectar el estado del conmutador del *cassette,* desconectar la ROM o direccionar bancos de RAM, etc.

14. El 6581. Es el SID *(Sound Interface Device).* Es el que proporciona al **64** sus increíbles capacidades sonoras.

15. El C64A es conocido también como **2218.** Se trata del controlador del bus serie.

idad



16, 17, 18. Estos *chips,* de 24 patillas, están bautizados como 901226-01, 901227-02 y 901225-01. Son los *chips* que contienen todo el código para el BASIC en ROM, y permiten programar al **64.**

19, 20. Existen dos **6526** en el **Commodore 64.** Son un dispositivo conocido por *Complex Interface Adapter* (CIA) y disponen de 40 patillas. La razón de que sean 2 es que uno maneja el SRQ y el otro las NMI. Estos son exactamente los nombres dados a un par de patillas del **6510,** que se utilizan para manejar las interrupciones.

21. Random Access Memory *(memoria de acceso aleatorio).* Sirven para un montón de cosas en el **64,** estando todo almacenado en estos *chips.* La cantidad de RAM disponible al conectar el ordenador es de 38.911 Kbytes, aunque podría incrementarse desconectando varios de los *chips* de ROM y conectando más RAM.

El **CBM 64** llegó en un momento dado, restándole protagonismo al **VIC-20.** Aunque el aspecto exterior de ambos resulta muy similar, las diferencias internas, en cuanto a prestaciones y diseño, son bastantes.

Suponemos que muchos lectores poseerán uno. Otros muchos no lo tienen y quizás estarán pensando en él. A estos segundos está orientado principalmente el artículo, aunque no estará de más que todos le echemos una ojeada.

El **CBM 64** está basado en el microprocesador de 8 bits **6510** de **MOS Technology,** firma que desarrolló el **6502,** utilizado en el **VIC-20.** Básicamente es un **6502,** al que se han incluido 8 líneas adicionales de Entrada/Salida (E/S), que se han revelado muy útiles de cara a la gestión de la memoria del **64.** Controlan la conmutación por bancos entre la ROM interna y los cartuchos de ROM externa o un dispositivo de E/S, y permite que cualquiera de ellos parezca estar ocupando una determinada parcela del espacio de direccionamiento del procesador. Esta conmutación por bancos proporciona al **64** un total de 84 Kbytes de memoria: 20K de ROM (memoria de solo lectura) y 64 K de RAM (memoria de acceso aleatorio). Los programas en BASIC pueden acceder a 39 Kbytes de la RAM y los programas en lenguaje máquina pueden acceder a 52 Kbytes de la RAM.

El **64** puede ser expandido de diversas formas. En la parte posterior de la carcasa existe un *slot* destinado a los cartuchos de programas y juegos. En el lado derecho hay dos *ports* para *joysticks* o *paddles* de juegos y un zócalo para insertar el conector de la fuente de alimentación.

El **64** dispone de un modulador interno de UHF, que permite utilizar el televisor como monitor de video. Todo lo que hay que hacer es conectar su salida con la entrada de antena del televisor, al conectar el **64** y sintonizar en torno al canal 36, aparece el mensaje de saludo del ordenador.

El **64** también dispone de un conector que proporciona salida de la señal compuesta de video, para atacar directamente un monitor, si se desea. La salida de señal de audio, que puede ser aplicada a un amplificador de alta fidelidad.

El ordenador dispone de un *port* serie, destinado a la conexión de una unidad de *diskettes* o la impresora.

El *interface* para conexión del *cassette* es otro *port* disponible. Por último, existe un *port* del usuario destinado a alojar otros dispositivos periféricos. De todas formas, no estaría mal que se hubieran previsto los *ports* **RS-232C,** en su versión estandarizada, o paralelo, tipo **Centronics,** para posibilitar la conexión de periféricos de otras marcas. Lo mismo podría haber ocurrido con las conexiones para el *cassette,* que inicialmente deber ser el modelo desarrollado por este mismo fabricante.

El chip **6567** es el *Video Interface Chip,* más conocido como **VIC-II,** y sirve para controlar la visualización en pantalla. Proporciona la generación del formato de 25 líneas de hasta 40 caracteres en modo texto, con los 16 colores posibles y el juego de caracteres gráficos. El *chip* produce también visualización de gráficos con mayor resolución, sobre una "rejilla" de 200 por 320 puntos, pudiéndose conseguir hasta 255 posibilidades de combinación de colores entre primer término y fondos. Asimismo puede soportar la existencia de un máximo de 8 *sprites* (espiritus), bloques móviles que simulan objetos, en la pantalla simultáneamente.

El **Commodore 64** tiene un *chip* llamado SID *(Sound Interface Device),* con nomenclatura **6581,** que produce notas musicales y efectos de sonido, cubriendo una gama de 9 octavas en hasta tres voces simultáneamente. Todos los aspectos de la generación de sonido puede ser controlados por medio de las instrucciones POKE del BASIC. Bajo control del usuario, programando por *software,* están los parámetros de forma de onda, forma de la envolvente del sonido y selección de las notas.

Casualmente, el **64** puede correr algunos de los programas desarrollados para el **PET,** aunque en nuestro país no hay un parque instalado demasiado grande de estos modelos.

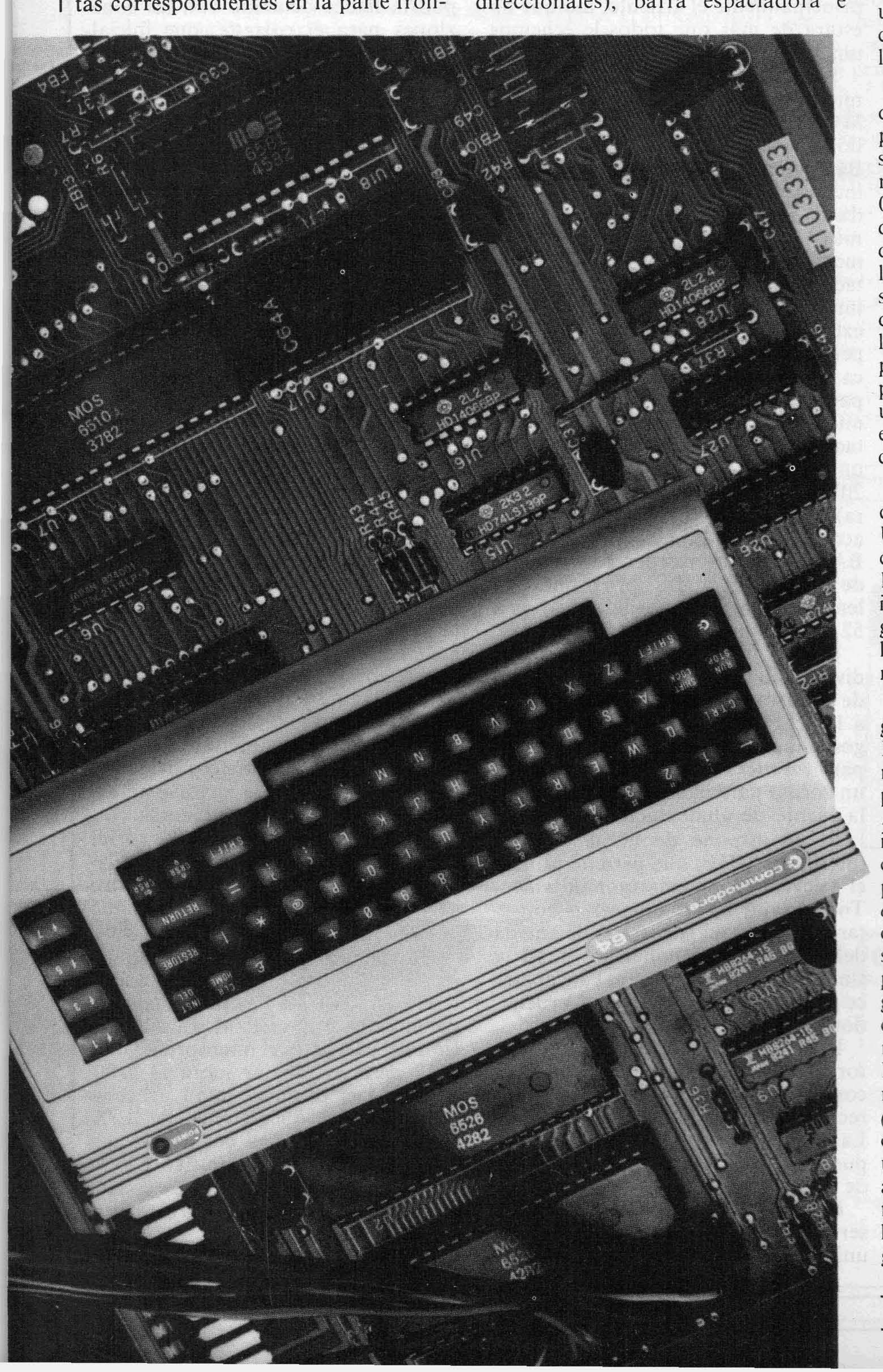
Opcionalmente, el **64** lleva emparejada una importante opción. Mediante el cartucho **Z-80** se implementa el sistema operativo **CP/M.** Con la adición del cartucho, el microprocesador **6510** pasa a trabajar junto al Z-80, actuando el primero como procesador para las operaciones de E/S, mediante las rutinas del **Kernel.** La ventaja del sistema operativo CP/M es la gran cantidad de *software* escrito para él, aunque eso sí, la mayoría con las leyendas en inglés.

El teclado tiene un total de 66

teclas, su configuración difiere bastante de lo que sería el estándar. Por ejemplo, las 4 teclas con 8 funciones programables (2 cada una) se sitúan en la parte derecha del conjunto, siendo definida su utilidad por el programa dado.

Los caracteres gráficos y las funciones especiales son accesibles desde el teclado, encontrándose las etiquetas correspondientes en la parte frontal de la tecla. Aunque el teclado resulta cómodo de manejar, los teclistas rápidos le podrían aducir alguna deficiencia, debido a que el *buffer* sólo tiene capacidad para almacenar 10 golpes de tecla.

Las únicas teclas que llevan incorporada la autorepetición con las correspondientes al movimiento del cursor (son 2 para los 4 movimientos direccionales), barra espaciadora e Insert/Delete, que inserta un espacio o borra un caracter, según se haya presionado la tecla SHIFT (mayúsculas) o no.



En el modo gráfico/texto, presionando la tecla SHIFT y una tecla alfabética se obtendrá el carácter gráfico mostrado en el lado derecho de la tecla.

Presionando la tecla **Commodore** y una tecla alfabética, se obtienen los caracteres gráficos que aparecen en el lado izquierdo de la tecla.

El **64** utiliza las 8 líneas adicionales de E/S del **6510** como *port* de control por mapa de memoria. Las 8 líneas son tratadas como una dirección de memoria, que reside en la dirección 0001. La dirección 0000 es un registro de control, que determina la dirección del flujo de datos que van por las líneas. Enviando un 0 al bit menos significativo (llamado bit 0) de la dirección 0001, se puede reemplazar la ROM del BASIC por RAM, en el proceso conocido como conmutación por bancos (de memoria). Enviando un 0 al bit 1 de la misma dirección, se elimina la ROM del **Kernal,** que contiene el sistema operativo del **64.**

La conmutación por bancos explica como el **64** puede correr **CP/M.** Utilizando la conmutación por bancos, para reemplazar las ROM por RAM, el sistema se queda con 64 K intactos, en parte de los cuales no guarda el **CP/M.** Naturalmente, esto lleva consigo la posibilidad de diseñar nuevos sistemas operativos propios.

El **64** proporciona diversos modos gráficos. Utilizando el comando POKE, se introducen valores en los 47 registros de control del *chip* **VIC-II,** para establecer los distintos modos. Aunque utilizar POKE puede ser incómodo al programar, se obtiene como contraprestación el trabajo rápido con los modos desde BASIC. El *chip* **VIC-II** está diseñado para acceder a 16 Kbytes de memoria, lo cual significa que, por ejemplo, se le puede programar para acceder a la ROM generadora de caracteres o a cualquier área de 16 K, dentro de la memoria. Utilizando instrucciones POKE y PEEK para introducir valores en el *port* A del *chip* CIA **6526** *(Complex Interface Adapter)*, se puede controlar qué bancos de memoria utilizará el **VIC-II.** Esta fijación libera al microprocesador **6510** de consumir tiempo en la tarea de controlar todos los gráficos, proporcionando al programador una herramienta flexible.

Cuando se conecta el **64** por vez primera, se establece automáticamente el modo gráfico en configuración de 25 líneas de 40 caracteres, pudiéndose utilizar 2 juegos de 256 caracteres. Cada caracter de los 2 juegos se forma a partir de una matriz de 8 por 8 puntos, que contiene letras mayúsculas y minúsculas, signos de puntuación, gráficos y otros símbolos especiales, incluyendo caracteres normales y en invierso.

Se eligió el formato de 25 por 40 debido a que el **64** se orientó para ser utilizado con el televisor doméstico, en el momento del diseño. Ocurre que la anchura de banda utilizada en los televisores es relativamente estrecha, pudiendo aparecer limitaciones en visualización de los gráficos si se utilizasen más caracteres por línea.

En principio, la pantalla visualiza caracteres en azul oscuro sobre un fondo azul claro. Presionando una de las teclas numéricas y la tecla de Control o la **Commodore,** se pueden cambiar los caracteres visualizados a uno de los 16 colores diferentes. El fondo y el borde también pueden ser cambiados por medio de comandos POKE. Por ejemplo, POKE 53280,x y POKE 53281,y (siendo x e y los números del color). Los colores de la pantalla pueden ser restaurados a sus valores iniciales presionando simultáneamente las teclas Run/Stop y Restore.

Cada una de las 1.000 posiciones destinadas en la pantalla para alojar un caracter, tiene una correspondencia con un byte en la memoria. Esto es lo que se conoce como visualización por mapa de memoria. Utilizando comandos POKE, se puede introducir un número de 8 bits en cada una de las localizaciones de memoria destinadas a caracteres. La ROM generadora de caracteres del **64** proporciona la estructura de puntos correspondientes al byte.

La posibilidad de definir caracteres propios parte de instruir al sistema para que busque los caracteres en una posición de memoria de la RAM, en lugar de la ROM, donde previamente hemos depositado el diseño.

La visualización en mapa de memoria también permite obtener gráficos en mapa de bits, esto quiere decir que cada punto de la pantalla corresponde a un bit de la memoria. Debido a que cada caracter está compuesto por un bloque de 8 bytes, el mapa de bits, en los 1.000 bloques de la pantalla, debe contener 8.000 bytes de memoria ó 640.000 bits. Cada uno de estos bits puede ser activado (encendido) o desactivado (apagado). La pantalla de alta resolución del **64** tiene un formato de 200 por 320 puntos, pudiéndose elegir 2 colores distintos para cada bloque de 8 por 8 puntos. Un modo gráfico multicolor permite una elección de 8 colores diferentes, pero el precio que se paga es reducir la resolución a 160 por 200 puntos.

Los *sprites,* en realidad objetos móviles por la pantalla, son caracteres definibles por el usuario en un formato de hasta 21 por 24 puntos.

Son generados y controlados por el *chip* **VIC-II,** que puede tomar cualquier forma en cualquiera de los 16 colores dados. Son idóneos para su utilización en juegos tipo marcianitos. Aunque son independientes de los gráficos normales, pueden ser utilizados desde el BASIC en programas. Cada *sprite* posee su propia localización de 63 bytes en la memoria y sus registros de posición y color propios.

Es posible activar un *sprite* introduciendo un 1 en el bit apropiado del registro de *sprite* del *chip* **VIC-II.** Después se programa su forma, se da el valor correspondiente al color y se indica su posición en la pantalla. Cada movimiento se define proporcionando cada vez un nuevo conjunto de coordenadas cartesianas.

También se puede ampliar el *sprite* hasta dos veces sus dimensiones originales, pero su resolución cae cuando se expande al máximo tamaño de 12 por 21 pares de puntos.

Cada *sprite* tiene una prioridad de visualización. Cuando los *sprites* cruzan sus trayectorias, uno parece que pasa por detrás del otro. Esto produce el efecto tridimensional en los gráficos.

El *chip* **VIC-II** igualmente puede detectar colisiones entre *sprites* o entre *sprites* y los datos del fondo. Para examinar los registros de colisión (*sprite* con *sprite* o *sprite* con datos) del **VIC-II,** se puede recurrir a sentencias PEEK. Cada *sprite* tiene un bit correspondiente en este registro, si el bit es 1, ese *sprite* está implicado en la colisión. La característica para detección de colisiones simplifica la programación requerida para diseñar cierto tipo de juegos.

El *chip* **SID,** aparte de las posibilidades de generación musical descritas anteriormente, posee algunos trucos más en la manga. El contenido armónico de una nota puede ser alterado mediante el filtraje de las señales. El **SID** dispone de 3 tipos diferentes de filtros, todos ellos clásicos: paso alto (sólo pasan las frecuencias altas), paso bajo (sólo pasan las frecuencias bajas) y paso banda (pasa una gama de frecuencias en torno a la frecuencia central). Con la ayuda de los filtros se puede controlar la forma de onda de los sonidos.

El **Kernal** es el nombre que recibe el sistema operativo utilizado en el **Commodore 64.** Su misión consiste en coordinar el funcionamiento correcto del sistema. Se encuentra depositado en una memoria ROM, grabada en fábrica, y se sitúa en las direcciones de memoria por encima de la RAM, entre las $EOOO y $FFFF.

Cuando entra en acción el **Kernal,** nada más conectar el ordenador, inmediatamente activa el BASIC.

El BASIC se encuentra en otra ROM, cuyas direcciones están comprendidas entre las $AOOO y $BFFF.

El BASIC empleado por el **64** es la versión 2 de una implementación propia de **Commodore** del BASIC estándar de **Microsoft.** El intérprete de este lenguaje de programación se ha incluido en 8 Kbytes de ROM. La extensión añadida con respecto a la versión de **Microsoft** se centra, principalmente, en comandos propios para el manejo de ficheros y las operaciones de E/S.

La mayoría de las instrucciones del BASIC pueden ser abreviadas en tan solo 2 letras. Un programa escrito utilizando esta técnica abreviada visualiza los comandos como una letra y un símbolo gráfico.

Al definir variables, el ordenador sólo utiliza las 2 primeras letras del nombre atribuido a la misma. Las líneas del programa están limitadas a un máximo de 80 caracteres.

Una de las características más importantes que tiene el **64** es su potente editor de pantalla. Una vez que se ha escrito el programa en BASIC, se puede mover el cursor a través de todo él, haciendo las correcciones precisas, por el simple método de

escribir encima de los caracteres en cuestión. Una línea modificada puede ser insertada en un programa solamente posicionando el cursor en esa línea y presionando la tecla Return. Es posible utilizar la tecla Insert/Delete para añadir o suprimir caracteres de una línea.

En adición, este ordenador presenta un peculiar modo "comillas" en el editor. Si el cursor queda posicionado a la derecha de un número impar de comillas, en la línea de un programa, se pueden introducir los códigos de control de cursor y de control del color dentro de las cadenas. Cuando es impreso en pantalla el texto incluido en la cadena, los códigos de cursor y color ejecutan automáticamente sus respectivas tareas. Esto proporciona al programador otra forma de controlar las visualizaciones en pantalla.

El almacenamiento de datos con el **Commodore 64** se puede llevar a cabo de 2 posibles maneras: con *cassette* o con unidades de *diskette.*

En el caso de utilizar *cassettes,* **Commodore** proporciona el modelo **1530 Datassette,** que lleva el conector adecuado para enchufar en la parte posterior del **64.** En su interior dispone del *interface* adecuado. Sucede que como la pantalla en el **64** es relocalizable, la memoria debe ser accedida cada cierto tiempo para su refresco. Cuando eso ocurre, el *interface* para cinta podría perder sus datos. Así que el **64** pone la pantalla en blanco mientras está corriendo la cinta. Cuando encuentra la cabecera del programa, para el motor durante unos segundos, pero esta pausa puede evitarse presionando la tecla **Commodore** cuando aparece el mensaje en pantalla.

La unidad de *diskettes* ofertada por **Commodore** es su modelo **VIC-1541,** un periférico inteligente, que goza de microprocesador propio para controlar las E/S, 2 Kbytes de RAM y su sistema operativo para disco (DOS) incorporado en 16 Kbytes de ROM. Utiliza *diskettes* de 5-1/4 pulgadas, pudiendo almacenar hasta 170 Kbytes de información en simple cara y simple densidad. El poseer el DOS en ROM simplifica las operaciones con disco y ahorra espacio en la memoria del ordenador y también en el disco. Igualmente evita la destrucción accidental del DOS, la única posibilidad sería si se destruyera voluntariamente la ROM.

Los comandos del DOS son extensiones de los mismos comandos empleados para el *cassette.* El **64** también soporta distintos tipos de ficheros en disco, a saber: secuenciales, aleatorios y relativos.

Se trata en definitiva de un excelente instrumento para iniciarse en el conocimiento del ordenador. No olvidemos que los ordenadores más potentes que se vendían en la década de los años 60, tenían 16 Kbytes de memoria por término medio, necesitaban una sala con aire acondicionado y su tamaño era mucho mayor que el de los actuales microordenadores, como el que hemos analizado en este artículo, además no necesitan tarjetas perforadas ni artilugios similares, simplemente se arreglan con la pantalla de un modesto televisor.

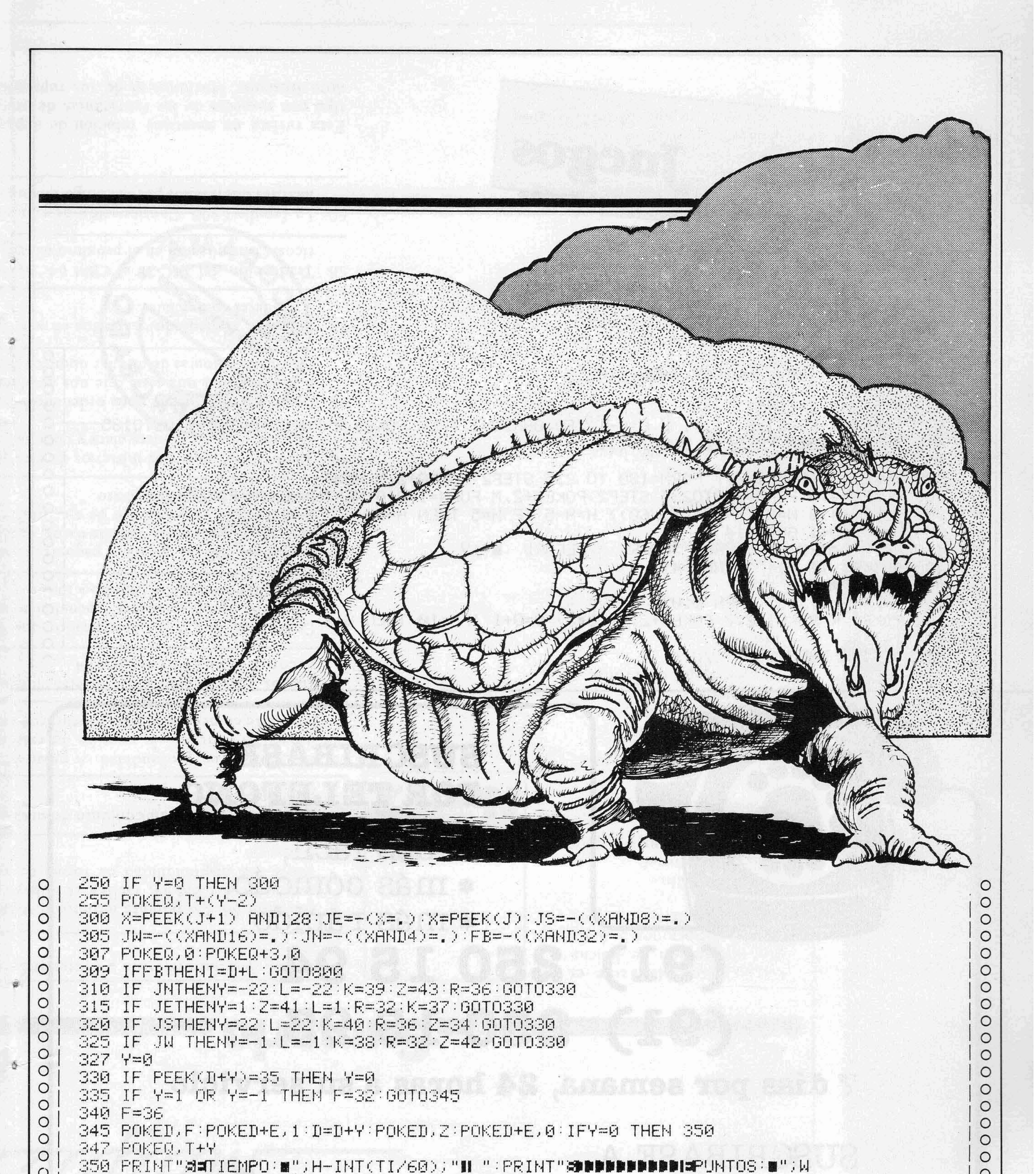
Anibal Pardo

VIC-20

El dragón temible

Con este programa el jugador se convierte en un dragón, dando vueltas por una cueva llena de corredores y escaleras, con el fin de alimentarse. El alimento, en forma de bicho, se mueve de forma elatoria por el interior de la cueva. Hay que seguirlo, moviéndose con el *joystick* y cuando esté al alcance hay que apretar el botón de disparo, con lo que el dragón lanza una llamarada que asa instantáneamente al animal, y es que los dragones no soportan la carne cruda. Además, hay que luchar contra el tiempo. Primero se dispone de sesenta segundos para comerse el primer bocado. Si se consigue, el jugador-dragón se apunta tantos puntos como segundos le restan. Entonces hay que ir a por otro bocado, pero esta vez sólo con cincuenta y cinco segundos de tiempo, el siguiente con cincuenta y así sucesivamente, hasta que sólo se dispone de diez segundos para zamparse la alimaña. Si lo consigue, el jugador vuelve a empezar con sesenta segundos. Como advertencia, el borde de la pantalla cambia de color cuando sólo quedan cinco segundos para que se acabe el tiempo. Si en el tiempo disponible no se consigue nada que llevarse a la boca, el juego termina.

```
10 REM***EL DRAGON TEMIBLE***
20 POKE 36879,233:PRINT"[illegible]EL DRAGON TEMIBLE"
30 POKE 52,28:POKE56,28:CLR
40 FORI=7168 TO 7679:POKEI,PEEK(I+25600):NEXT
50 POKE 36869,255
60 FORC=7432 TO 7551: READA:POKEC,A:NEXT
65 DATA 8,42,28,127,28,42,8,0,8,127,93,28,127,73,28,28
70 DATA 187,0,238,0,187,0,238,0,127,65,127,65,127,65,127,65
80 DATA85,255,85,0,0,0,0,0,170,255,170,0,0,0,0,0,28,8,28,8,28,8,28,8,8,28,8,28,8
,28,8
90 DATA 28,3,3,18,58,126,252,72,108,192,192,72,92,126,63,18,54,28,28,73,127,28,9
3,127,8
95 DATA0,56,124,230,198,206,124,56,68,170,170,146,170,68,186,0,32,16,108,254,254
,254
96 DATA 124,40
97 DATA 255,255,255,255,255,255,255,255
99 PRINT"J":POKE 36879,233:X=7701:FORT=1 TO 23:POKEX,47:POKEX+30720,1:X=X+22:NEX
T
100 X=7724:FORT=1TO21:POKEX,35:POKEX+30720,2:X=X+1:NEXT
105 X=7766:FORT=1TO 19:POKEX,35:POKEX+30720,2:X=X+22:NEXT
110 X=7746:FORT=1TO19:POKEX,35:POKEX+30720,2:X=X+22:NEXT
115 X=8164:FORT=1TO21:POKEX,35:POKEX+30720,2:X=X+1:NEXT
120 X=7768:FORY=1TO6
125 FORA=1TO2:FORT=1TO21:POKEX,35:POKEX+30720,2:X=X+1:NEXTT:X=X+1:NEXTA
130 X=X+22:NEXTY
135 X=7747:FORY=1TO7
140 FORT=1TO19:POKEX,36:POKEX+30720,1:X=X+22:NEXTT
145 X=X+(-415):NEXTY
170 V=0
180 X=0:H=60:POKE7701,47:POKE7723,47:POKE7701+30720,1:POKE7723+30720,1
185 POKEQ+2,0
190 TI$="000000":D=8152:Z=43:Y=-22:P=0:T=190:N=130:E=30720:L=-22:O=0:K=39:R=36
195 S=7756:J=37151:POKEJ+3,255:POKEJ+3,127:G=0:P=0:POKE36879,233:Q=36874:N=15:PO
KEQ+4,N
200 A=INT(RND(1)*4)+1:B=INT(RND(1)*3)+1:B=B*3
215 IF A=1 THEN M=+22:C=44:G=36
220 IF A=2 THEN M=-22:C=44:G=36
225 IF A=3 THEN M=-1:C=46:G=32
230 IF A=4 THEN M=+1:C=45:G=32
235 IF PEEK(S+M)=35 THEN 200
245 POKES,G:POKES+E,1:S=S+M:POKES,C:POKES+E,4
```



```
250 IF Y=0 THEN 300
255 POKEQ,T+(Y-2)
300 X=PEEK(J+1) AND128:JE=-(X=.):X=PEEK(J):JS=-((XAND8)=.)
305 JW=-((XAND16)=.):JN=-((XAND4)=.):FB=-((XAND32)=.)
307 POKEQ,0:POKEQ+3,0
309 IFFBTHENI=D+L:GOTO800
310 IF JNTHENY=-22:L=-22:K=39:Z=43:R=36:GOTO330
315 IF JETHENY=1:Z=41:L=1:R=32:K=37:GOTO330
320 IF JSTHENY=22:L=22:K=40:R=36:Z=34:GOTO330
325 IF JW THENY=-1:L=-1:K=38:R=32:Z=42:GOTO330
327 Y=0
330 IF PEEK(D+Y)=35 THEN Y=0
335 IF Y=1 OR Y=-1 THEN F=32:GOTO0345
340 F=36
345 POKED,F:POKED+E,1:D=D+Y:POKED,Z:POKED+E,0:IFY=0 THEN 350
347 POKEQ,T+Y
350 PRINT"■■TIEMPO:■";H-INT(TI/60);"▮ ":PRINT"■▶▶▶▶▶▶▶▶▶▶■PUNTOS:■";W
352 PRINT"■▶▶▶▶▶▶■RECORD :■";V
355 IF TI/60>=HTHEN400
360 IF TI/60>=H-5THEN POKE 36879,238
365 B=B-1:IF B=0 THEN 200
370 GOTO 215
400 POKEQ+4,0:POKED,F:POKED+E,1:POKES,G:POKES+E,1:IFW>V THEN V=W
410 PRINT"■   JUEGO  TERMINADO                    ":FORT=1TO1000:NEXT
```

Juegos

PROGRAMA ★ commodore Magazine ★ COMPROBADO

Viene de la página anterior

```
808 GOTO800
810 IF PEEK(I)=35 ANDO=0 THEN 310
812 I=I-L
814 POKEQ+2,T+(O*5)
815 POKEI,R:POKEI+E,1:I=I-L:O=O-1:IFO=0ANDU=1 THEN POKED,R:POKED+E,1:U=0:GOTO185
816 IF O=0 THEN POKEQ+2,0:GOTO310
818 GOTO 814
900 POKEI,33:POKEI+E,7:FORM=180 TO 235 STEP2:POKEQ+2,M:NEXT
901 POKEQ+4,N:FORM=180TO235 STEP2:POKEQ+2,M:FORN=1TO10:NEXTN:NEXTM:POKEQ+2,0
902 POKEQ+4,0:W=W+(H-INT(TI/60)):H=H-5:IF H=5 THEN H=60
904 U=1:O=O+1:GOTO814
420 PRINT"■■PULSA J PARA JUGAR     RECORD :■";V
430 GET A$:IF A$="" THEN 430
440 PRINT"■                                          ":GOTO180
800 IF PEEK(I)=35 THEN 810
802 POKEI,K:POKEI+E,7:POKEQ+2,T+(O*5):O=O+1:IFO=10 THEN 814
804 I=I+L
806 IF I=S THEN 900
```



VIC-20

Arácnido

A este programa quisimos llamarlo, en un principio, araña; pero claro, nos encontramos con el problema de que el **VIC-20** no incluye, entre sus caracteres estándar, la letra ñ. Por ello, al final, se quedó con el nombre de **Arácnido** donde, evidentemente, no hay que incluir ninguna ñ. Esto nos da pie para decir que pronto incluiremos alguna rutina que permita modificar el juego de caracteres y poder disponer así de la letra ñ, muy útil para palabras como España, año, araña... En cuanto al programa en sí se trata de un juego en el que el jugador, supuestamente convertido en araña debe comerse cuantas moscas pueda. Las moscas se mueven según un algoritmo que las aleja de la araña, y que a veces, hace que se muevan aleatoriamente. La araña, al moverse, va dejando tras de sí un rastro, producido al rozar el suelo, y que debe evitar, pues el programa está concebido de tal forma, que si la araña vuelve a pisar su rastro, el juego termina. La araña debe evitar también salirse de la pantalla, pues con ellos también termina el juego.

Una particularidad es que la mosca, al moverse, puede borrar la traza dejada por la araña en algunos puntos, permitiendo que la araña pase de nuevo por ahí sin pisar su rastro. Esto que parece tan complicado explicado sobre el papel es sencillo, una vez puestos a ello, con los dedos sobre las teclas. El programa dice qué teclas hay que utilizar, y lleva la cuenta de los puntos conseguidos. En definitiva, se trata de un programa sencillo y entretenido de arañas que atrapan moscas, aunque lo mismo podría ser de zorros atrapando conejos, porque lo importante es que usted se lo pase bien jugando con su ordenador.

```
1 REM*** ARACNIDO ***
2 REM
3 REM  VIC-20 SIN EXPANSION
4 REM
5 REM
10 GOTO1010
15 REM**ACTUALIZA PUNTOS**
20 PRINT"♥REC"; TAB(8-LEN(STR$(HS)));HS; "  PUNT.";TAB(22-LEN(STR$(SC)));SC
25 RETURN
30 REM**MOVIMIENTO HOMBRE**
40 GET K$
50 IF K$="I" THEN D=1
52 IF K$="J" THEN D=7
54 IF K$="K" THEN D=3
56 IF K$="M" THEN D=5
60 X=X+XD(D):Y=Y+YD(D):IF X<0 OR X>LL-1 OR Y<1 OR Y>LD OR PEEK(TL+Y*LL+X)=PL THE
```

Juegos

Viene de la página anterior

```
N 710
70 POKE TL+Y*LL+X,PL:POKE AT+Y*LL+X,RED
80 IF X<>SX OR Y<> SY THEN 90
82 FOR I=1TO5:SC=SC+1:GOSUB 20:FORJ=230 TO 250
84 POKE VL,15: POKES1,J:NEXTJ:NEXTI:POKEV,0:POKES1,0
86 SX=FNR(LL):SY=FNR(LD):IF PEEK(AT+SY*LL+SX)=RED THEN 86
87 POKE TL+SY*LL+SX,SP:POKE AT+SY*LL+SX,GREEN
90 REM**MOVIMIENTO DEL ARACNIDO**
100 DS=SQR((X-SX)↑2 +(Y-SY)↑2)/15:M=ABS(RND(1)>DS)+ABS(RND(1)>DS)
110 NX=SX+M*SGN(SX-X):NY=SY+M*SGN(SY-Y)
113 IF M=0 THEN 40
115 IF NX>1 AND NX<LL-1 AND NY>2 AND NY<LD-11 AND PEEK(AT+NY*LL+NX)<> RED THEN 1
30
120 NX=SX-XD(FNR(8)):NY=SY-YD(FNR(8)):M=M-1:GOTO 113
130 POKE AT+SY*LL+SX,1:SX=NX:SY=NY:POKE TL+SY*LL+SX,SP:POKE AT+SY*LL+SX,GREEN
140 GOTO 40
700 REM**FIN DEL JUEGO**
710 POKE VL,15:POKES1,230:FOR I=1 TO 200:POKES1,240:FORI=1TO 400:NEXT I
720 POKES1,245: FOR I=1 TO 500:NEXTI:POKES1,0
730 PRINT"███████▓▓▓▓▓▓▓PUNTOS";SC
740 IFSC>HS THEN FORI=128 TO 255 STEP3:POKEVL,15:POKES1,I:FORJ=1TOISTEP10:NEXTJ:
NEXTI
750 IF SC>HS THEN HS=SC:PRINT"████NUEVO RECORD"
760 POKES1,0:FOR I=1 TO 1000:NEXTI
800 REM**TITULO**
810 PRINT"♫":GOSUB 20
815 PRINT:PRINT:PRINT"    ███ARACNIDO███"
820 PRINT:PRINT"ENGULLE CUANTOS INSEC-TOS PUEDAS ANTES DE   CAER EN TU PROPIA TE
LA"
830 PRINT:PRINT"J=IZQDA. K=DCHA.        I=ARRIBA M=ABAJO"
840 PRINT:PRINT"BUENA SUERTE..."
850 PRINT:PRINT" PULSA E PARA EMPEZAR":PRINT"        S PARA SALIR"
870 I=AT
875 GET A$:IF A$="S" THEN PRINT"♫":END
877 IF A$="E" THEN 910
890 POKEI,FNR(6)+1
895 I=I+1:IFI< 38750 THEN 875
897 GOTO 870
900 REM**NUEVO JUEGO**
910 SC=0
915 PRINT"♫":GOSUB20
920 X=10:Y=10:D=1:SX=FNR(LL):SY=FNR(LD)
940 POKE TL+LL*Y+X,PL:POKEAT+LL*Y+X,RED:POKE TL+LL*SY+SX,SP:POKEAT+LL*SY+SX,GREE
N
990 GOTO 40
1000 REM**INICIO**
1010 PRINT"♫":POKE 36879,27
1020 TL=7680:AT=38400:LL=22:LD=22
1030 VL=36878:S1=36874:NS=36877
1040 PL=102:SP=81:RED=2:GREEN=5
1080 DEFFNR(X)=INT(RND(1)*X+1)
1100 FOR I=1 TO 8:READ XD(I),YD(I):NEXTI
1110 DATA 0,-1,1,-1,1,0,1,1,0,1,-1,1,-1,0,-1,1
1120 GOTO810
```



Commodore 64

Bombardeo en la ciudad

Con este juego para el **64** la barra de espacios del teclado se convierte en el botón para lanzar bombas desde un bombardero.

El bombardero sobrevuela una ciudad llena de altos rascacielos, sobre los que hay que dejar caer la mortífera carga. Para ello, el avión va efectuando, una tras otra, una serie de pasadas sobre la ciudad. En cada una el jugador dipone de un número de bombas, que puede elegirse a voluntad entre una y seis bombas. Desde luego que con más bombas se puede acabar antes con todos los edificios. El caso es que, para hacer el juego más interesante, en cada pasada, el avión vuela más bajo que en la anterior.

El programa requiere del jugador buenos reflejos para lanzar la bomba justo en el momento adecuado, ni antes ni después. Cada edificio destruido va sumando puntos, lo que se refleja en la pantalla, junto con el récord y la puntuación anterior.

```
10 REM***BOMBARDEO EN LA CIUDAD***
100 GOSUB 1000
200 GOSUB 2000:GOSUB710:POKE 54296,15
210 FOR J=1 TO 30
220 FOR T=20 TO RND(1)*10+(9-LEV)+5 STEP -1
230 POKE 1024+J+40*T,160:POKE 55296+J+40*T,5
240 NEXTT,J
249 REM***BUCLE PRINCIPAL***
250 FOR B=7 TO 19:H=LEV+1:FOR A=0 TO 39:O=A+40*B
260 GET A$
270 IF A$=" " THEN GOSUB 500
275 IF PEEK(1025+O)=160 THEN 4000
280 POKE 1023+O,32:POKE 55295+O,6
290 POKE 1024+O,252:POKE 55296+O,6
300 POKE 1025+O,98:POKE55297+O,6
320 NEXTA:GOSUB 400:NEXT B
330 GOSUB 800:GOTO210
399 REM***BORRA AVION***
400 POKE1023+O,32:POKE 55296,2
440 RETURN
499 REM***RUTINA DE DISPARO***
500 H=H-1:IF H<1 THEN RETURN
505 FORQ=B+2 TO 19:W=A+40*Q
510 IF PEEK(1024+W)=160 THEN GOSUB 600
520 POKE 1024+W-40,32:POKE 55256+W,2
530 POKE 1024+W,30:POKE 55296+W,2
540 NEXT Q
550 RETURN
599 REM***DESTRUCCION DE EDIFICIO***
600 S=S+10
610 PRINT"※■PUNTOS*";S
620 POKE 54276,129
630 POKE 54272,HF(Q-7):POKE 54273,HL(Q-7)
640 FOR T=1 TO 10:NEXT:POKE54276,0
650 RETURN
660 REM***PUNTUACION***
699 REM***PUNTUACION***
710 PRINT"※■PUNTOS*";S;TAB(20)"■RECORD*";H(1)
```

Juegos

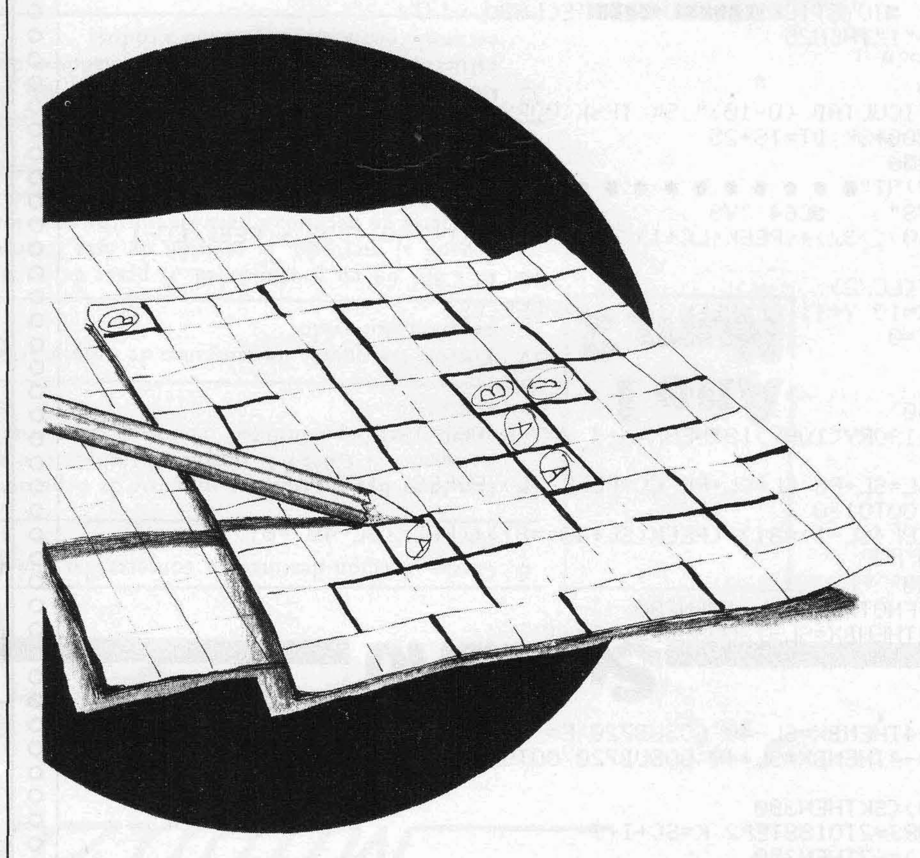
Viene de la página anterior

```
720 PRINT"■■■";TAB(10)"ULTIMA PUNTUACION";L
730 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■";TAB(6);"■CAMPEON■ "  ;H$(1)
735 PRINT TAB(8)"■BOMBAS POR PASADA■"     ;LEV
740 RETURN
799 REM***NUEVA PANTALLA***
800 POKE54276,17
805 FORU=3TO7STEP2:POKE54272,LF(U):POKE54273,HF(U):GOSUB870:NEXT
810 FORU=7TO3STEP-2:POKE54272,LF(U):POKE54272,HF(U):GOSUB870:NEXT
820 PRINT"■":GOSUB 710
830 POKE54276,0
860 RETURN
870 FORUP=1TO200:NEXT
880 RETURN
999 REM***VARIABLES***
1000 DIMH(10),H$(10),HF(20),HL(20)
1010 FORT=54272TO54285:READK:POKET,K:NEXT
1020 FORT=1TO20:READHF(T),HL(T):NEXT
1030 FORK=1TO10:H$(K)="EL ORDENADOR"
1040 NEXT
1090 RETURN
1999 REM***JUGAR OTRA VEZ***
2000 L=S:S=0
2080 POKE53280,7:POKE53281,7:PRINT"■"
2085 GOSUB6000:PRINT"■"
2090 RETURN
3999 REM***FIN RUTINA DE JUEGO***
4000 POKE1024+0,102:POKE55296+0,2:POKE54276,129
4010 POKE43272,75:POKE54273,34:FORYU=1TO500:NEXT:POKE54276,0
4040 PRINT"■":POKE53280,4:POKE53281,7
4050 FORI=1TO10:IFS>H(I)THENGOSUB4500:GOTO4070
4060 NEXT
4070 PRINT"■";TAB(12)"■■10 MEJORES"
4080 PRINT"■■"
4090 FORO=1TO10:PRINTTAB(5)O;TAB(10)H(O);TAB(20)H$(O):NEXT
4100 PRINT"■■■   PULSA F1 PARA EMPEZAR   "              "
4110 GETR$:IFR$<>"■"THEN 4110
4120 GOTO200
4499 REM***NUEVO NOMBRE***
4500 FORC=9TOISTEP-1:H(C+1)=H(C):H$(C+1)=H$(C):NEXT
4510 PRINT TAB(10)"■■■■■BIEN HECHO"
4520 PRINT"■■■■ESTAS ENTRE LOS 10 MEJORES"
4530 PRINT"■■■■■■■■■■■■ESCRIBE TU NOMBRE"
4540 INPUTH$(I):H$(I)=LEFT$(H$(I),19)
4560 H(I)=S:I=10
4570 RETURN
5999 REM***TITULO***
6000 REM
6010 PRINT"■":F$="BOMBARDEO":LEV=1
6020 FORUI=1TO37:PRINTTAB(UI)"■ ■■■"
6030 IFUI>14THENPRINTTAB(15)"■";LEFT$(F$,UI-14)
6035 FORGH=1TO 10
6040 NEXT GH,UI:PRINT"■■";TAB(38)"  "
6045 PRINT TAB(7)
6050 PRINT"■■";TAB(12)"■BOMBAS POR PASADA"     ;LEV
6060 PRINT"■■■■■■■■■■■■■■DESTRUYE LOS EDIFICIOS"
6070 PRINT"■■■■■■■■■■DISPONES DE POCAS BOMBAS"
6080 PRINT"■■■BARRA DE ESPACIOS PARA SOLTAR BOMBAS"
```

```
6090 PRINT"▮▮▮▮▮▮▮▮F1 CAMBIA EL NO. DE BOMBAS"
6095 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮PUEDES ESTRELLARTE"
6100 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮FELIZ BOMBARDEO"
6110 PRINT"▮▮▮▮▮▮▮▮▮▮PULSA ESPACIO PARA EMPEZAR■"
6120 GETD$
6130 IF D$=" " THEN RETURN
6140 IF D$<>"■"THEN6120
6150 LEV=LEV+1:IFLEV=7 THEN LEV=1
6160 PRINT"▮▮▮▮▮▮";TAB(12)"BOMBAS POR PASADA";LEV:GOTO6120
10000 DATA0,0,0,0,129,255,255,0,0,255,15,65,255,255
10010 DATA7,53,8,23,8,147,9,159,10,205,11,114,12,216,14,107,16,47,17,37,19,63
10020 DATA 21,154,22,227,25,177,28,214,32,94,34,75,38,126,43,52,45,198
```

Commodore 64

Punto y raya



Pocos estudiantes han podido resistir la tentación de coger un papel y un bolígrafo y ponerse a jugar con un compañero a uno de esos clásicos juegos que todos conocemos.

Aquí os presentamos, es uno de ellos, aunque en este caso tu contrincante será el ordenador.

El objeto de "Punto y raya" es hacer que tu contrincante dibuje el tercer lado de un cuadrado y así tú puedes poner el cuarto anotándote un punto a tu favor. Cada vez que termines uno te toca de nuevo poner otra raya. El ordenador además de ser tu enemigo es el árbitro, no te permitirá hacer trampa y... no te preocupes porque él tampoco te la hará.

Dispone de diez niveles de dificultad, a mayor nivel, más tiempo tardará en responder el **Commodore** y más difícil te será ganarle. Además, si dispones de *joystick* puedes jugar con él.

Una vez esté el programa en la memoria del ordenador, al ejecutarlo, lo primero que te preguntará será si vas a hacer uso del teclado o del *joystick*, a lo que deberás responder con T o J, respectivamente.

Después deberás seleccionas el nivel de dificultad. Si eliges el cero, jugará aleatoriamente, por lo que te resultará fácil ganarle; ni te imagines que va a ocurrir algo parecido si eliges el diez. Hecho esto, aparecerá en pantalla el tablero de juego, consistente en diez filas de diez puntos negros cada una, y un marcador doble que te indicará el tanteo entre el **64** y tú.

Para jugar conduce el cursos amarillo hasta donde quieras dibujar una raya (si juegas con el teclado, mediante las teclas de movimiento del cursor), y pulsa el disparador de tu *joystick* o bien RETURN si usas el teclado.

Cuando el **64** hace un movimiento,

Juegos

Viene de la página anterior

produce un *flash* intermitente en la raya que él va a pintar para que veas dónde está moviendo.

En el instante en que cierres un cuadrado, éste se colorea de "CYAN" y te toca de nuevo.

Cuando el **64** completa uno, lo rellena de rojo y él mueve otra vez.

El juego continúa hasta que se forman 81 cuadritos y el ordenador declara quién es el vencedor.

Al copiar la línea 90 usa la forma abreviada del comando PEEK, esto es, teclea P y a continuación SHIFT/E, pues si no, no te cabrá la línea dentro de la longitud permitida (80 caracteres).

Si no te gusta la forma de controlar el movimiento del cursor con el teclado, puedes asignárselo a las teclas que desees con sólo cambiar el carácter entrecomillado para realizar los movimientos hacia: "arriba", "derecha", "abajo" e "izquierda", respectivamente en las líneas 950, 960, 970 y 980.

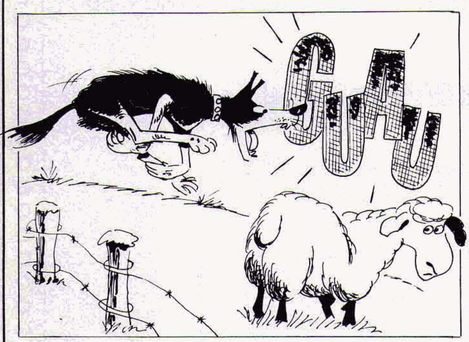
Si dispones de *joystick*, conéctalo en el Port 1.

```
0 REM ***PUNTO Y RAYA***
5 POKE53281,12:POKE53280,0
10 PRINT"■■■■"SPC(14)"PUNTO Y RAYA
20 PRINT"■■                    ■JOYSTICK■■■■■■■■■■■■TECLADO (J/T) ?";
25 GETA$:IFA$<>"J"ANDA$<>"T"THEN25
26 PRINTA$
30 JK=-1:IFA$="T"THENJK=0
40 INPUT"■■■■NIVEL DE DIFICULTAD (0-10)";SK:IFSK<0ORSK>10THEN40
50 SK=(10-SK)/10:TS=200-200*SK:DT=TS+25
60 SC=1024:CO=13*4096+8*256
70 PRINT"■":FORI=1TO10:PRINT"■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■■":NEXT
80 YS=0:VS=0:PRINT"■TU:"YS"     ■C64:"VS
90 DEFFNBX(LC)=(PEEK(LC+40)<>32)+(PEEK(LC+1)<>32)+(PEEK(LC-40)<>32)+(PEEK(LC-1)<
>32)
100 DEFFNVH(LC)=LC<>2*INT(LC/2)
110 SL=SC+450:CL=CO+450:X=19:Y=11:CC=PEEK(SL):CR=PEEK(CL)
120 POKESL,160:POKECL,7:F=0
130 IFJKTHEN150
140 GOSUB930:GOTO160
150 GOSUB860:IFPC=0THEN150
160 X=X+J:Y=Y+K:IFX<1ORX>19ORY<1ORY>19THENX=X-J:Y=Y-K:GOTO130
170 IFPC=99THEN200
180 POKESL,CC:POKECL,CR:SL=SL+PC:CL=CL+PC:CC=PEEK(SL):POKESL,160
190 CR=PEEK(CL):POKECL,7:GOTO130
200 L=(PEEK(SL+1)=81)+(PEEK(SL-1)=81)+(PEEK(SL+40)=81)+(PEEK(SL-40)=81)
210 IFL+(CC=32)=-3THEN230
220 I=128:GOSUB710:GOTO130
230 WH=1:ML=SL:GOSUB650:IFNOTFNVH(SL)THEN280
240 IFX>1ANDFNBX(SL-1)=-4THENBX=SL-1:GOSUB720:F=-1
250 IFX<19ANDFNBX(SL+1)=-4THENBX=SL+1:GOSUB720:GOTO110
260 IFFTHEN110
270 GOTO310
280 IFY>1ANDFNBX(SL-40)=-4THENBX=SL-40:GOSUB720:F=-1
290 IFY<19ANDFNBX(SL+40)=-4THENBX=SL+40:GOSUB720:GOTO110
300 IFFTHEN110
310 WH=2:F=0:CN=0:IFRND(0)<SKTHEN390
320 FORI=80TO720STEP80:FORJ=2TO18STEP2:K=SC+I+J
330 IFPEEK(K)=32ANDFNBX(K)=-3THEN350
340 NEXTJ,I:GOTO390
350 I=K:IFPEEK(I-40)=32THENI=I-40:GOTO540
360 IFPEEK(I+40)=32THENI=I+40:GOTO540
370 IFPEEK(I-1)<>32THENI=I+1:GOTO600
380 IFPEEK(I+1)<>32THENI=I-1:GOTO600
390 I=INT(INT(RND(0)*20)*40+21*RND(0)+SC+41):CN=CN+1:IFPEEK(I)<>32THEN390
```

```
400 IFNOT((PEEK(I+1)=81ANDPEEK(I-1)=81)OR(PEEK(I+40)=81ANDPEEK(I-40)=81))THEN390
410 IFSK<>.6ORCN>TSTHEN470
420 IFFNVH(I)THEN450
430 IFFNBX(I-40)=-2ORFNBX(I+40)=-2THEN390
440 GOTO540
450 IFFNBX(I-1)=-2ORFNBX(I+1)=-2THEN390
460 GOTO600
470 IFFNVH(I)THEN510
480 IFSK<>.6ORCN>DTTHEN540
490 IFFNBX(I+40)=-2ANDFNBX(I-40)=-2THEN390
500 GOTO540
510 IFSK<>.6ORCN>DTTHEN600
520 IFFNBX(I+1)=-2ANDFNBX(I-1)=-2THEN390
530 GOTO600
540 ML=I:GOSUB650
550 IFFNBX(ML-40)=-4THENBX=ML-40:GOSUB720:F=-1
560 IFFNBX(ML+40)=-4THENBX=ML+40:GOSUB720:GOTO310
570 IFFTHEN310
580 GOTO110
590 IFNOT(PEEK(I-40)=81ANDPEEK(I+40)=81)THEN390
600 ML=I:GOSUB650
610 IFFNBX(ML-1)=-4THENBX=ML-1:GOSUB720:F=-1
620 IFFNBX(ML+1)=-4THENBX=ML+1:GOSUB720:GOTO310
630 IFFTHEN310
640 GOTO110
650 CL=CO+ML-SC
660 POKEML,67
670 IFFNVH(ML)THENPOKEML,93
680 I=185:IFWH=2THENI=150
690 FORJ=1TOWH:POKECL,0:GOSUB710:POKECL,1
700 FORL=1TO200:NEXT:POKECL,0:NEXT
710 FORK=1TO200:NEXT:RETURN
720 YS=YS+1:J=3:I=200:CL=CO+BX-SC:IFWH=2THENJ=2:I=150:YS=YS-1:VS=VS+1
730 POKEBX,160:POKECL,1
740 FORL=1TO3:POKECL,J:GOSUB710:POKECL,1:FORK=1TO200:NEXT:I=I+18:POKECL,J:NEXT
750 PRINT"SQQQQQQQQQQQQQQQQQQQQQQQ ■TU:"YS"     ■C64:"VS
760 IFYS+VS<81THENRETURN
770 PRINT"SQQQQQQQQQQQQQQQQQQQQQQ":IFYS>VSTHEN800
780 PRINT"■         PERDISTE         "
790 GOTO850
800 PRINT"■GANASTE !!!"
850 END
860 J=0:K=0:PC=0
870 I=PEEK(56321)
880 IF(IAND1)=0THENPC=-40:K=-1:RETURN
885 IF(IAND8)=0THENJ=1:PC=1:RETURN
890 IF(IAND2)=0THENPC=40:K=1:RETURN
900 IF(IAND4)=0THENPC=-1:J=-1:RETURN
910 IF(IAND16)=0THENPC=99
920 RETURN
930 J=0:K=0:PC=0
940 GETA$:IFA$=""THEN940
950 IFA$="■"THENPC=-40:K=-1:RETURN
960 IFA$="■"THENJ=1:PC=1:RETURN
970 IFA$="■"THENK=1:PC=40:RETURN
980 IFA$="■"THENJ=-1:PC=-1:RETURN
990 IFASC(A$)=13THENPC=99
1000 RETURN
```

PROGRAMA ★ commodore Magazine ★ COMPROBADO

Concurso



Este programa nos lo envía **José Antonio Serrano Bergali,** de Sevilla. El programa es para **VIC-20,** bien en versión estándar o con ampliación de memoria de 3K. El juego consiste, como su nombre indica, en hacer de perro pastor, conduciendo a su redil a 10 alborotadoras ovejas, que no siempre están muy dispuestas a obedecer al perro. El can es controlado mediante cuatro teclas, A y Z para el movimiento vertical y «y» para el horizontal.

Cuando el perro se acerca lo suficiente a una de las ovejas, ésta sale corriendo parándose después de un breve trayecto. El movimiento de la oveja es aleatorio, pero presenta un cierto sesgo en la dirección del redil. Hay que ir empujando a todas y cada una de las ovejas hasta que las 10 estén a buen recaudo, con lo que termina el juego. El programa lleva un contador de tiempo, así que para obtener el título de buen pastor hay que ser rápido manejando las ovejas.

```
100 REM**PERRO PASTOR*
110 POKE650,255:POKE36878;15
120 GOSUB880
130 TI$="000000"
140 PRINT"♥######################";
150 FORO=0TO20
160 PRINT"#                    #";
170 NEXTO
180 PRINT"######################";
190 FORO=7732TO7842STEP22:POKEO,35:POKEO+30720,6:NEXTO
200 POKE7836,35:POKE7837,35:POKE7839,35:POKE7840,35
210 FORO=1TO10
220 A=INT(RND(1)*306)+7856
230 IFPEEK(A)=32THENPOKEA,15:NEXTO:GOTO250
240 GOTO220
250 A=INT(RND(1)*306)+7856:IFPEEK(A)=32THENPOKEA,16:GOTO270
260 GOTO250
270 IFPEEK(7729)=15THEN740
280 GETA$
290 IFPEEK(A)=15ORPEEK(A-1)=15ORPEEK(A+1)=15ORPEEK(A-22)=15ORPEEK(A+22)=15THEN40
0
300 IFPEEK(A-21)=15ORPEEK(A-23)=15ORPEEK(A+21)=15ORPEEK(A+23)=15THEN400
310 IFPEEK(A-1)=35THENIFA$=","THEN270
320 IFPEEK(A+1)=35THENIFA$="."THEN270
330 IFPEEK(A+22)=35THENIFA$="Z"THEN270
340 IFPEEK(A-22)=35THENIFA$="A"THEN270
350 IFA$=""THEN270
360 IFA$="A"THENPOKEA,32:A=A-22:POKEA,16:GOTO270
370 IFA$="Z"THENPOKEA,32:A=A+22:POKEA,16:GOTO270
380 IFA$=","THENPOKEA,32:A=A-1:POKEA,16:GOTO270
390 IFA$="."THENPOKEA,32:A=A+1:POKEA,16:GOTO270
400 IFPEEK(A-1)=15THENC=A-1:GOTO480
410 IFPEEK(A+1)=15THENC=A+1:GOTO480
420 IFPEEK(A-22)=15THENC=A-22:GOTO480
430 IFPEEK(A+22)=15THENC=A+22:GOTO480
440 IFPEEK(A-21)=15THENC=A-21:GOTO480
450 IFPEEK(A-23)=15THENC=A-23:GOTO480
460 IFPEEK(A+21)=15THENC=A+21:GOTO480
470 IFPEEK(A+23)=15THENC=A+23:GOTO480
480 L=INT(RND(1)*7)+1
```

PROGRAMA ★ commodore Magazine ★ COMPROBADO

PREMIADO CON 5.000 PESETAS

```
490 FORO=1TOL
500 POKEC,32:POKE36876,232:POKE36876,0
510 B=INT(RND(1)*7)
520 IFB=0ORB=4THENC=C-22
530 IFB=1ORB=5ORB=6THENC=C-1
540 IFB=2THENC=C+22
550 IFB=3THENC=C+1
560 IFPEEK(C)=15ORPEEK(C)=35ORPEEK(C)=16THENGOSUB610
570 POKEC,15
580 GOTO660
590 NEXTO
600 GOTO270
610 IFB=0ORB=4THENC=C+22
620 IFB=1ORB=5ORB=6THENC=C+1
630 IFB=2THENC=C-22
640 IFB=3THENC=C-1
650 RETURN
660 FORX=7813TO7819STEP3:IFPEEK(X)=15THEN690
670 NEXTX:IFPEEK(7709)=15THENX=7709:GOTO690
680 GOTO590
690 FORG=1TO10:POKEX,15:FORY=1TO100:NEXTY:POKEX,32:FORY=1TO100:NEXTY:NEXTG
700 FORX=7703TO7707:IFPEEK(X)=32THENPOKEX,15:GOTO270
710 NEXTX
720 FORX=7725TO7729:IFPEEK(X)=32THENPOKEX,15:GOTO270
730 NEXTX
740 POKE36869,240
750 PRINT"■■■■■HAS TARDADO:"
760 PRINT"■■"MID$(TI$,3,2)" MINUTOS"
770 PRINT"■■"MID$(TI$,5,2)" SEGUNDOS"
780 IFTI$<"000300"THENPRINT"■■■■■■ERES EXCELENTE PASTOR":GOTO810
790 IFTI$<"000500"THENPRINT"■■■■■■ERES UN BUEN PASTOR":GOTO810
800 IFTI$>"000500"THENPRINT"■■■■■■ERES UN MAL PASTOR"
810 PRINT"■■■■[PULSA P PARA SEGUIR]"
820 PRINT"■[PULSA F PARA ACABAR]"
830 GETA$:IFA$="P"THEN860
840 IFA$="F"THENPRINT"■■■■■ENTONCES.. HASTA OTRA.":END
850 GOTO830
860 POKE36869,254:GOTO130
870 STOP
880 PRINT"■■■■■■■■■PERRO PASTOR"
890 PRINT"■■■■■■■■■■■POR:"
900 PRINT"■■■■J.ANTONIO SERRANO"
910 PRINT"■■■■■■■■■■■■■■■[21/2/84]"
920 POKE36869,254
930 FORI=6144TO6704:POKEI,PEEK(I+26624):NEXTI:FORI=0TO2000:NEXTI
940 PRINT"■■ EL JUEGO CONSISTE EN  ■METER LAS OVEJAS EN  ■ SU REDIL."
950 PRINT"■■PARA ELLO POSEES A TU FIEL PERRO QUE TE":PRINT"■■■■■AYUDARA."
960 PRINT"■ DEBES HACERLO EN EL   ■MENOR TIEMPO POSIBLE":PRINT"■■A:ARRIBA  Z
ABAJO"
970 PRINT"<:IZQUIERDA  >:DERECHA"
980 FORI=6705TO7679:POKEI,PEEK(I+26624):NEXTI
990 PRINT"■■■■■■[PULSA TECLA]"
1000 GETA$:IFA$=""THEN1000
1010 PRINT"■"
1020 FORI=6264TO6279:READF:POKEI,F:NEXTI
1030 RETURN
1040 DATA0,32,108,126,63,30,20,20
1050 DATA0,64,195,254,124,60,36,36
```

Concurso

Un programa de conversión decimal-hexadecimal y viceversa, para transformar números de una base a otra con toda sencillez. Lo envía **M. Soler,** de Madrid. El programa se maneja a través de un pequeño menú con tres opciones. Conversión hex-decimal, decimal-hex y salida del programa. Para transformar un número de una base a otra, basta con escribir el número de partida y pulsar RETURN, el resultado aparece en pantalla.

Después de cada conversión se puede cambiar de opción, volviendo al menú, o se puede permanecer en la opción inicial. El programa viene escrito en BASIC, sin referencias a memoria, por lo que se puede ejecutar tanto en el **VIC-20** como en el **64.**

```
10 REM M.SOLER * CONVERSOR HEX.<>DEC.
20 POKE53280,7:POKE53281,15:PRINT"▒▒"CHR$(8):POKE775,0:POKE809,255
40 PRINT"▒▒     CONVERSOR HEXADECIMAL <> DECIMAL     ▒▒"
50 PRINT" OPCIONES:▒"
60 PRINT" ▒1▒ - PASAR DE HEXADECIMAL A DECIMAL.▒"
70 PRINT" ▒2▒ - PASAR DE DECIMAL A HEXADECIMAL.▒"
80 PRINT" ▒3▒ - TERMINAR.▒▒"
90 PRINT"▒OPCION ELEGIDA ?"
100 GETOPE$:IFOPE$=""THEN100
110 OPE=VAL(OPE$)
120 IFOPE<1OROPE>3THEN100
130 ONOPEGOTO140,300,500
140 PRINT"▒▒▒    CONVERSOR  HEXADECIMAL > DECIMAL     ▒▒"
150 INPUT"NUMERO HEXADECIMAL: ";HEX$
155 LHE=LEN(HEX$)
170 FORJ=LHETO1STEP-1
180 CI=ASC(RIGHT$(HEX$,J))
190 IFCI<48OR(CI>56ANDCI<65)ORCI>70THENGOSUB1000:GOTO140
200 CI=CI-48:IFCI>8THENCI=CI-7
210 TA=CI*(2↑(J*4-4))
220 RE=RE+TA
230 NEXT
240 PRINT"▒▒"HEX$" EN DECIMAL ES: "RE
250 INPUT"▒▒▒▒▒▒▒▒▒▒VAS A CAMBIAR DE OPCION ";RES$
260 IFLEFT$(RES$,1)="N"THENRUN140
270 IFLEFT$(RES$,1)<>"S"THEN250
280 RUN
300 PRINT"▒▒▒    CONVERSOR  DECIMAL > HEXADECIMAL     ▒▒"
310 INPUT"NUMERO DECIMAL: ";DEC$
320 LDE=LEN(DEC$):DEC=VAL(DEC$)
330 FORM=1TOLDE:C=ASC(MID$(DEC$,M,1))
340 IFC<48ORC>57THENGOSUB1000:GOTO300
345 NEXT
350 GOTO370
360 LDE=LDE-1
370 BI=DEC/(2↑(LDE*4-4))
375 IFBI<>INT(BI)THENDEC=DEC-(2↑(LDE*4-4))*INT(BI):GOTO380
377 DEC=0
380 IFINT(BI)>9THENBI$=CHR$(BI+55):GOTO390
385 BI$=CHR$(BI+48)
390 RE$=RE$+BI$
395 IFLDE=1THEN420
400 GOTO360
420 IFLEFT$(RE$,1)<>"0"THEN440
430 RE$=RIGHT$(RE$,LEN(RE$)-1):GOTO420
440 PRINT"▒▒"DEC$" EN HEXADECIMAL ES: "RE$
450 INPUT"▒▒▒▒▒▒▒▒▒▒VAS A CAMBIAR DE OPCION ";RES$
460 IFLEFT$(RES$,1)="N"THENRUN300
470 IFLEFT$(RES$,1)<>"S"THEN470
```

PROGRAMA ★ commodore Magazine ★ COMPROBADO



```
480 RUN
500 POKE53280,14:POKE53281,6:POKE809,246:PRINT"]]"CHR$(9);:END
1000 PRINT"QQQ"TAB(11)"R NO ES CORRECTO "
1010 FORI=0TO1000:NEXT
1020 PRINT"]
1030 RETURN
```

●●●

Carta de colores es un programa de utilidad que nos envía **Fernando-José García** de Barcelona para el **Commodore 64.** El programa, al ejecutarse, va presentando una a una todas las posibles combinaciones de colores para el cuadro y el borde de la pantalla, al mismo tiempo que presenta los valores de las posiciones de memoria que controlan cuadro y fondo, correspondientes a cada combinación.

Cuando se halla una combinación interesante, se pueden anotar sus valores para utilizarlos en cualquier momento. Las combinaciones se pueden ver despacio o deprisa, es posible repetirlas y además el programa aconseja sobre el color más adecuado de los caracteres, para cada combinación.

```
0 REM*RUN90PARA SOLO COMBINACION*
1 PRINT"]QQQQQQQQQQQQ      CARTA DE COLORES":POKE53280,8:POKE53281,7:PRINT
2 PRINT"]______________________________________________"
5 PRINT"Q"
6 PRINT"QQQAPUNTE LOS NUMEROS QUE DESEE EN UN PAPEL"
7 FORI=1TO5000:NEXTI
8 PRINT "]":INPUT"1=RAPIDO,2=LENTO";V
9 IFV=1THENVE=300
10 IFV=2THENVE=2000
20 FORBA=0TO15:FORBO=0TO15 .
30 POKE53280,BA
40 POKE53281,BO
50 PRINT"]QQS-BORDE..POKE53280,"BA
60 PRINT"QQ-FONDO..POKE 53281,"BO
70 FORX=1TOVE:NEXTX
80 NEXTBO:NEXTBA
90 PRINT"Q":PRINT"]":INPUT"QQQ*DESEA REPETIR ALGUNA COMBINACION";R$
95 IFR$="SI"THEN100
98 IFR$<>"SI"THENEND
110 INPUT"QQ  COLOR DEL FONDO..";F
120 POKE53280,B
125 POKE53281,F
130 PRINT"QQQ COLOR SUGERIDO DE LETRA"
131 PRINT"___________________________________"
135 IFF=0THENPRINT" 2,4,5,6,8,9,11,13,14,15"
136 IFF=1THENPRINT" 5,6,7,10,11,15"
137 IFF=2THENPRINT"4,5,8,9,11,13,15,16"
138 IFF=3THENPRINT"3,6,7,10,12"
139 IFF=4THENPRINT"3,7,12,16"
140 IFF=5THENPRINT"1,4,7,8,10,12"
141 IFF=6THENPRINT"4,5,8,13,14,15,16"
142 IFF=7THENPRINT"3,7,9,10,12,13"
143 IFF=8THENPRINT"3,7,8,9,12,16"
144 IFF=9THENPRINT"4,5,6,8,9,11,13,14,16"
145 IFF=10THENPRINT"1,2,3,7,8,10,12"
146 IFF=11THENPRINT"4,5,6,8,9,11,13,14,15,16"
147 IFF=12THENPRINT"1,3,4,7,8,10,12,13,16"
148 IFF=13THENPRINT"3,5,7,10,12"
149 IFF=14THENPRINT"1,3,4,7,12,16"
150 IFF=15THENPRINT"3,5,6,7,9,10,12,13,15"
154 PRINT"QQQ**PULSE-ESPACIO-PARA OTRA COMBINACION**"
155 GETA$:IFA$<>" "THEN155
160 PRINT"]":GOTO100
```

Base de datos para el CBM-64

Base de datos es un término que todos los iniciados en la informática habrán escuchado alguna vez. Tras él se esconde lo que podría ser, por ejemplo, una agenda telefónica, una serie de datos bibliográficos o algo similar. Digamos que se podría definir como un fichero, en el cual se guardan distintas informaciones, siguiendo un determinado tipo de estructura. Se trata de algo similar a lo que podríamos hacer con una simple libreta de notas. La principal diferencia estriba en que el soporte sobre el que escribimos no es papel, sino todo lo contrario. La información se almacenaría en cinta o disco magnético pudiendo posteriormente ser procesados en la memoria central del ordenador para un sinfin de distintas posibilidades.

La base de datos por sí sola carece de interés, si no va acompañada por otro concepto: el sistema de gestión de la base de datos. Muchas veces las publicaciones hacen referencia a él por sus siglas inglesas **DBMS** *(Data Base Management System).* En resumidas cuentas, no es otra cosa que el *software* que se encarga de acceder a los datos previamente grabados. Su principal cualidad reside en permitir que el usuario se olvide totalmente de todas las cuestiones que tengan que ver con el almacenamiento de los datos en forma física, sea en disco o cinta. Es decir, gestiona el manejo y funcionamiento de los ficheros.

De cara a comprender lo que queríamos decir con estructura de los datos almacenados, nos valdremos de un par de ejemplos. Una agenda personal de direcciones incluiría datos tales como el nombre, apellido, dirección, teléfono, ocupación, etc. de cada persona incluida, escritos siempre en un determinado orden (dejando en blanco los datos que no tengamos).

El ejemplo de un fichero bibliográfico, los datos se almacenarían en forma de título del libro, nombre del autor, palabras que resuman la temática abordada, etc. Sigue también una estructura previamente establecida, de tal manera que el sistema de gestión conozca en qué orden y manera va a encontrar los datos de cualquier información que busque.

Principalmente, tres son las clases de base de datos existentes, dependiendo de la manera en que se organicen los datos: jerárquicas, en red y relacionales.

Las relacionales son las más modernas y más sencillas de manejar por el usuario. Por ejemplo, en un archivo bibliográfico, se podrían localizar todas las obras de un autor determinado, que en su título lleven una palabra dada, o su temática trate sobre algo concreto.

A partir de la base de datos, previamente creada, se pueden manejar los datos para reordenarlos de varias posibles maneras, seleccionando los que interesan y pasándolos al papel en un variado número de formas.

¿QUE MAS PUEDE HACER UNA BASE DE DATOS?

Una buena Base de Datos, debe proporcionar al usuario una serie de funciones como:

— La posibilidad de definir al formato de los ficheros, cuantos registros va a contener cada fichero, como van a aparecer los registros en la pantalla, cuantos campos va a tener cada registro, cuál va a ser la longitud de los campos (cuántos caracteres va a tener cada campo).

— Después de definir el formato del fichero, se ha de poder introducir la información en los distintos campos, creándose así el fichero por primera vez.

— En cualquier momento, la base de datos debe ofrecer la posibilidad de modificar la información almacenada, permitiendo añadir o eliminar registros o campos reemplazando, añadiendo o borrando los datos necesarios.

— La base de datos, y ésta es una característica importante, debe per-

mitir la búsqueda selectiva de registros en los ficheros, según diversos criterios, de modo que sea sencillo encontrar cualquier registro con sólo recordar alguno de los datos que contiene. Por ejemplo, se debe poder encontrar el registro de una persona, recordando su nombre, o su teléfono, o quizás sólo el número de su calle.

— Por último, es interesante que la base de datos, permita generar informes impresos con los datos del fichero, organizados en forma de tablas, calculando totales y subtotales, y con los títulos y textos descriptivos necesarios para una adecuada presentación.

SUPERBASE 64

Superbase 64 es un Sistema de Gestión de Bases de Datos, desarrollado para el **Commodore 64**, por la compañía de *software* **Precisión Software Limited.** El programa viene en *diskette* y le acompaña un manual, traducido al castellano, y que pretende servir tanto de manual de aprendizaje, a través de diversos ejemplos de construcción y manejo de los ficheros, como de posterior guía de referencia o recordatorio de los comandos, una vez que el usuario se ha familiarizado con el manejo y utilización del programa.

Cómo empezar

La primera consideración a tener en cuenta es la del equipo *hardware* necesario. **Superbase 64,** necesita, como elementos imprescindibles, un **Commodore 64,** una unidad de discos y, al menos, un *diskette* para almacenar datos, además del *diskette* del **Superbase.** El uso de una impresora, aunque no es imprescindible, se hace necesario para poder aprovechar un conjunto de funciones que permiten generar e imprimir informes, con los datos almacenados.

Disponiendo del equipo necesario, se puede proceder a cargar el **Super base,** para ello, no hay más que introducir el *diskette* en la unidad de discos, y escribir en pantalla LOAD "SB", 8, 1 pulsando RETURN. Al cabo de unos 2 minutos, el programa estará cargado. Si es la primera vez que se utiliza el **Superbase**, hay que crear un disco de datos, para ello, basta con introducir un *diskette* virgen en la unidad de discos y pulsar la tecla F1. El programa se encarga de formatear el disco, copiando a continuación en el mismo, una serie de pantallas de ayuda, con información sobre las funciones más utilizadas. Si no es la primera vez que se utiliza ya se habrán creado algunos discos de datos, y entonces, basta con introducir el disco de datos que se desea utilizar y pulsar RETURN para acceder a los comandos del **Superbase.**

Estructura de los datos y acceso a los comandos

Toda la información se guarda en *diskettes.* Según la terminología empleada en el manual, cada *diskette* constituye una base de datos, a la que hay que asignar un nombre y un identificador de 2 cifras. Los datos se almacenan de la forma usual en este tipo de programas, es decir, agrupados en ficheros, cada uno de los cuales está constituido por una serie de registros idénticos, dentro de los cuales existen un conjunto de campos, o zonas donde se guardan los datos. El **Superbase** permite almacenar en cada diskette hasta 15 ficheros diferentes, cada uno de los cuales puede estar constituido por tantos registros como sean necesarios. Por otro lado, el número de bases de datos (o *diskettes*) que pueden crearse es ilimitado, en definitiva, una buena cantidad de espacio para guardar todo tipo de datos.

```
mode : menu              Superbase 64
V 1.0E   (c) Precision Software,1983

File Selected  = materiales

     f1  File
     f2  Format
     f3  Batch
     f4  Sort
     f5  Prog
     f6  Maintain
     f7  Memo
     f8  Help
```

```
mode : menu              Superbase 64
V 1.0E   (c) Precision Software,1983

File Selected  = materiales

     f1  Enter
     f2  Select
     f3  Find
     f4  Output
     f5  Calc
     f6  Report
     f7  Execute
     f8  Help
```

Para acceder a las distintas funciones de manejo de ficheros, **Superbase** ofrece dos posibilidades, la más sencilla es a través del uso de menús. Posee dos menús principales que dan acceso a 15 funciones diferentes, además, existen varios menús secundarios, relacionados con alguna de las funciones principales, a los que se accede al seleccionar dicha función. Puede verse el aspecto y opciones de los menús en las figuras que acompañan el artículo. La otra forma de acceder a las funciones del **Superbase** es escribiendo directamente los comandos a ejecutar en una línea de la parte superior de la pantalla, denominada línea de comandos. Esta posibilidad es mucho más flexible que la anterior, pues permite realizar secuencias de comandos uno tras otro, sin más que pulsar una tecla, y lleva a uno de los aspectos más interesantes que es la posibilidad de escribir programas mezclando los comandos de base de datos y sentencias BASIC.

Diseño de ficheros

El primer punto a considerar, en un Sistema de Gestión de Bases de Datos, es la facilidad y posibilidades de diseño de los ficheros por parte del usuario. La opción FORMAT del **Superbase**, permite que el usuario diseñe la configuración de cada fichero, como más le guste. Cuando se entra en esta opción, para diseñar un nuevo fichero, el cursor aparece en una pantalla en blanco. Mediante las teclas de movimiento del cursor, el usuario puede moverse por la pantalla, y proceder a diseñar lo que podríamos llamar una "ficha en blanco". Para ello escribirá los nombres de los campos donde guardará información, definiendo asimismo la longitud de cada campo (cuántos caracteres va a contener) y, el tipo de información que posteriormente introducirá" en cada campo (textos, fechas, números, resultados de cálculos). También en este punto, se puede escoger la combinación de colores para el fondo, el reborde, y el texto, que se considere más agradable. Además, si la pantalla queda corta al diseñar la ficha, se pueden utilizar hasta 3 pantallas más para la misma ficha siendo accesibles mediante las teclas + y -.

Después de borrar, modificar y añadir campos, o nombres de campos, y cuando el diseño sea satisfactorio, se puede proceder a guardarlo en el disco. De esta forma se ha creado la "ficha modelo", todas las fichas o registros del fichero serán idénticas a ella, salvo, claro está, que la información dentro de los campos será diferente para cada una. Antes de pasar a ver cómo se introducen los datos, es interesante señalar, que la opción FORMAT permite modificar ficheros en cualquier momento, aunque ya contengan información, lo cual puede resultar de gran utilidad cuando el fichero se queda pequeño y se desea añadir nuevos campos, o simplemente se decide cambiar el formato de los registros por otro diferente.

Entrada de datos y modificaciones en los mismos

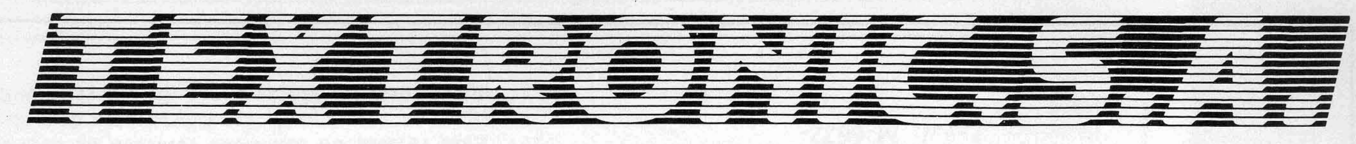
La introducción y actualización de datos es la función que más corrientemente se utiliza en una base de datos. Para ello, el **Superbase** proporciona la opción ENTER. Una vez diseñada la configuracin de las fichas o registros del fichero, se puede introducir en ella la información deseada. Al seleccionar la opción ENTER, en la pantalla aparece una ficha en blanco, idéntica a la que el usuario creó con la opción FORMAT. El cursor aparece situado al principio del primer campo del registro. Lo único que hay que hacer es ir escri-

```
mode :  Select Menu

f1  Key
f2  Current
f3  Next
f4  Last
f5  Previous
f6  First
f7  Match
f8  Output
A   Add
R   Replace
D   Delete
```

```
mode :  Maintain Database

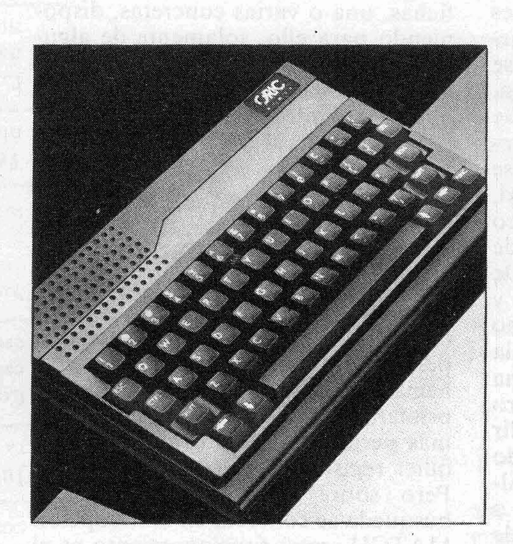
f1  Status
f2  Catalog
f3  Import
f4  Export
f5  Directory
f6  Backup
f7  New Disk
f8  Other
```



SU CASA EN INFORMATICA

En el centro de Madrid la mejor exposición en ordenadores personales

Venga a visitarnos. Disfrute de nuestro salón dispuesto para que usted y sus hijos puedan utilizar todos y cada uno de nuestros ordenadores



personales, destinados a que usted compruebe su utilidad y sus hijos aprendan a divertirse con sus juegos preferidos

Damos cursillos prácticos para que usted maneje en breve su ordenador.

Los mejores precios en HARDWARE Y SOFTWARE las mejores marcas: ORIC, COMMODORE, LASER, DRAGON, SINCLAIR y todos los juegos en el mercado

Estamos en la calle Preciados n.° 39 (Plaza de Santo Domingo) MADRID Tfno. 248 56 35

Si quiere más información escríbanos sin compromiso y nos pondremos en contacto con usted.

Nombre ....

Apellidos ....

Domicilio ....

Localidad .... D.P. ....

Provincia ....

biendo la información en el campo, pulsando RETURN al terminar, con lo que el cursor pasa al campo siguiente. De esta forma se va introduciendo la información hasta completar todos los campos. Salvo en un campo, que se denomina campo de claves, escogido por el usuario cuando diseña el registro, y en el que es obligatorio introducir alguna información, cualquier otro campo se puede dejar en blanco si así se desea. Después de llegar al último campo, se puede almacenar la ficha en el *diskette*. El **Superbase** preguntará si se quiere llenar otra ficha, siendo así, presentará una nueva ficha en blanco en la pantalla, que podrá rellenarse de la misma manera que la anterior. De esta forma, se pueden ir llenando y almacenando tantos registros como se quiera, sin más limitación que la capacidad del *diskette*. Esta misma opción se utiliza con cualquier fichero ya existente, en el que se desea añadir más información, bien sea añadiendo nuevos registros, o modificando alguno de los existentes, añadiendo o borrando información en alguno de sus campos. Existen otros comandos como ADD, REPLACE o DELETE, que permiten añadir, reemplazar o borrar registros.

La posibilidad de búsqueda selectiva de registros en un fichero, es una de las características que definen la potencia de una base de datos, y es la que mejor refleja las ventajas de su empleo es frente a los sistemas de ficheros manuales. Pensemos, por ejemplo, en lo que supone buscar, en un fichero de cientos o miles de fichas, una o varias concretas, disponiendo para ello, solamente de algunos datos aislados. El proceso manual se lleva a cabo en minutos o quizás segundos, según el tamaño del fichero y las características del programa.

Las posibilidades que a este respecto ofrece el **Superbase**, se concretan en tres comandos: SELECT, FIND y SORT.

El comando SELECT, que dispone de su propio menú de subopciones, permite moverse por todo el fichero, hacia adelante o hacia atrás, desde el primero hasta el último registro. Además permite el acceso rápido a cualquier registro especificando su clave. Pero sobre todo, su potencia de búsqueda se concreta en la subopción MATCH, cuyo funcionamiento es el siguiente: al escoger esta opción, aparece en la pantalla un registro en blanco, es decir, con los nombres de los campos, pero sin ninguna información en su interior. Mediante las teclas de cursor, se puede entrar en cualquiera de los campos y escribir la información que se recuerde, por ejemplo, se podría entrar en un campo de nombres y especificar el nombre, o el apellido, o sólo la inicial del apellido. Hecho esto, basta con pulsar SHIFT + RETURN con lo que aparecerá en pantalla el primero de los registros con la información especificada. Pulsando M aparecerá el siguiente registro que coincida con los datos suministrados y así sucesivamente.

Existen varias formas de especificar los criterios de selección, que hemos resumido en el cuadro adjunto.

FIND es otro comando similar a MATCH, pero que crea y almacena una lista de las claves de los registros seleccionados. Esta lista puede ser utilizada posteriormente. La forma de especificar los criterios de selección es la misma que con el comando MATCH.

Por último, la opción SORT permite reordenar un conjunto de registros o todos los del fichero, en orden alfabético, pero según el contenido de otros campos distintos del campo clave y que hay que especificar. También crea y almacena una lista de estos registros reordenados, lista que podrá utiliarse posteriormente en

CUADRO DE CRITERIOS DE SELECCION

"Texto" es la información recordada de algún campo

TIPO DE CORRESPONDENCIA	COMO SE ESPECIFICA
Exacta	"= texto"
Deslizante dentro de campo	"texto"
Exacta exclusiva (distinto que)	"#texto"
Deslizante del campo en adelante	"texto-"
Mayor que	"<texto"
Menor que	">texto"
Rango de valores	"<tl&>t2"
De texto parcial	"= texto *"
Con carácter variable	"= texto??"

Todos estos modos pueden combinarse para buscar un registro.

```
mode : Entry                          :#  1   k

Titulo <                                       >

Autor  <                                        >

Tema   <                                        >

Editorial <                                     >

Fecha de edicion  <        >

Dimensiones <           > No.Pag.<              >

Precio  <        >
```

cualquier momento. Por ejemplo, se puede generar con ella informes impresos, o utilizarla en un programa.

Informes impresos

El **Superbase** posee unas capacidades, para la generación de informes impresos, que la sitúan por encima de muchos otros programas de gestión de Bases de Datos. A través de dos opciones, OUTPUT y REPORT, es posible imprimir una gran variedad de informes. Se puede utilizar para ello cualquier lista de registros previamente seleccionados, o todos los registros del fichero. Se puede imprimir información de todos los campos de cada registro o sólo de algunos. La información puede presentarse en forma de tablas, o hileras y pueden incluirse, en cualquier punto del informe, todo tipo de textos explicativos, títulos, encabezamientos. Es posible formatear la información, empezando a imprimir en cualquier fila o columna y cualquiera de los símbolos del teclado se puede usar para subrayados o marca de separación. La estructura del informe se puede almacenar endisco para ser utilizada en cualquier momento. Además, mediante las funciones CALC y BATCH, es posible efectuar cálculo entre campos de uno o varios registros, permitiendo calcular e imprimir totales y subtotales como parte del informe.

Programación

Esta aplicación ofrece la posibilidad de escribir programas, utilizando todo el conjunto de comandos de manejo de ficheros, sentencias BASIC, y una serie de comandos específicos, que incluyen sentencias condicionales y formatos de impresión, como márgenes, número de líneas, longitud del papel.

Los programas se escriben como líneas de comandos numeradas consecutivamente, y pueden ser almacenados en el *diskette* para posterior utilización. Esta opción, amplía las posibilidades del **Superbase**, permitiendo a cada usuario construir programas que se ajusten exactamente a sus necesidades. Se dispone de hasta 4K de memoria para la confección de programas, lo que es más que suficiente para la mayoría de los casos, y existen algunas facilidades, no suficientemente aclaradas en el manual, para establecer enlaces entre diversos ficheros de una Base de Datos.

Mantenimiento

La última opción que consideramos es la MAINTAIN, que también posee un menú propio de subopciones, y que permite una serie de facilidades para el mantenimiento de las bases de datos. Estas facilidades incluyen el formateo de nuevos *diskettes* de datos, la copia de seguridad (back-up) de datos previamente existentes, la posibilidad de importar y exportar información entre los ficheros secuenciales, así como la capacidad de visualizar listas y catálogos de los ficheros y programas contenidos en cada *diskette.*

Conclusiones y veredicto

Trabajar con el **Superbase** puede ser frustrante en algunas ocasiones, pues el programa no incluye información del tipo "que hacer a continuación". Por ejemplo, a veces se pregunta qué fichero se va a utilizar, pero no hay una forma sencilla de obtener una lista de los ficheros existentes. Los menús no están numerados, ni llevan encabezamientos para identificarlos. Para salir de una opción, a veces hay que pulsar **Q**, mientras que otras veces es la tecla RUN/STOP,

```
mode  Select n,l,p,f,k,c,m,o,a,r,d,+-

Titulo <Los propios dioses          >

Autor  <Isaac Asimov                >

Tema   <Ciencia Ficcion             >

Editorial <Bruguera                 >

Fecha de edicion  <NOV1980>

Dimensiones <17.5x10.5 > No.Pag.<   331>

Precio  <      200>
```

```
mode : Entry                          #  1   k

No.Fact<          > Fecha<        >

Apellidos<                              >
Nombre  <                   >

Direccion<                              >
Poblacion<             >
Provincia<             >

              Telf.<          >

Importe  <          >  Desc.<    >

Total<          >
```

pero el programa no lo dice. El manual se muestra ambigüo en una serie de apartados, por ejemplo, en cómo crear ficheros secuenciales para conectar con un programa de tratamiento de textos, siendo pocos los ejemplos concretos de programación, y los de uso de ficheros secuenciales.

También se echa en falta la posibilidad de crear ficheros que sean una subdivisión de ficheros existentes. Podría hacerse con un programa pero no es nada sencillo.

A pesar de estos pequeños fallos, el **Superbase** es un programa muy completo, que proporciona una amplia gama de facilidades, capaces de convertir el **Commodore 64** en un pequeño microordenador de gestión.

●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●

Algunos ejemplos de utilización de superbase

Vamos a ver, con algunos ejemplos concretos, algunas de las posibilidades que ofrece el **Superbase.** Más que desarrollar, paso a paso, un ejemplo de como se construye y maneja un fichero, desarrollo que puede encontrarse en el manual de aprendizaje, quizás sea más interesante mostrar algunos de los resultados que se pueden obtener, tanto en pantalla como a través de la impresora, en cuanto al diseño de ficheros, generación de listas y reordenación de ficheros.

Diseño

Las posibilidades de diseño de ficheros, se concretan, en el **Superbase,** en la opción FORMAT. Con ella se pueden diseñar los registros de cada fichero, especificando los nombres de los campos donde se guardará información, el tipo de campo, que se corresponde con el tipo de información que va a contener (textos, fechas, valores numéricos), y su longitud, es decir, cuantos caracteres va a poder almacenar cada uno.

Un primer ejemplo de diseño, puede ser la creación de un fichero bibliográfico, en el que guardemos información sobre un conjunto de libros. Para cada libro habrá una ficha (o registro) en el fichero, con información sobre: Tema del libro, autor, título, editorial, fecha de la edición, n.° de páginas, dimensiones y precio de compra. Empleando la opción FORMAT realizamos el diseño del registro, en el que habrá campos de texto (para el autor, título, etc...), campo de fechas y campos numéricos, para el número de páginas y el precio. El resultado es el registro de la fig. 1, tal y como aparece en la pantalla.

Con la opción ENTER, se puede introducir información en cuantos registros se quiera, es decir, se pueden almacenar datos de tantos libros como se quiera introduciendo la información de cada libro en un registro diferente. En la fig. 2, se puede ver el resultado correspondiente a un registro del fichero, en el que la información ya ha sido depositada.

Otro ejemplo, en el que interviene un nuevo tipo de campo, puede ser un fichero de facturas de clientes, en una supuesta empresa. El diseño del fichero aparece en la fig. 3, incluye campos de texto, de fechas y numéricos, pero lo más interesante es el último campo, TOTAL, que es un campo de resultados. Ello quiere decir, que el valor de dicho campo no es necesario introducirlo, sino que es el propio programa el que lo obtiene, como resultado de cálculos efectuados con otros campos del registro. En nuestro caso, el contenido del campo TOTAL se obtiene como el contenido de IMPORTE menos el contenido de DESCUENTO especificado en tanto por ciento. Así pues, en el campo TOTAL hay que especificar la fórmula:

```
 mode : Processing
Press Space to Enter Another

No.Fact<53870      > Fecha<08FEB84>

Apellidos<Gomez Garcia                >
Nombre   <Manuel              >

Direccion<c/Mateo Inurria    n 589    >
Poblacion<Madrid              >
Provincia<Madrid              >

               Telf.<9231500   >

Importe <       53894> Desc.<    5>

Total<      51199>
```

```
Compre Jupiter              Bruguera
El americano impasible      Bruguera
Los propios dioses          Bruguera
Otra vuelta de tuerca       Bruguera
Primavera negra             Bruguera
```

Madrid del 4 al 28 de Abril

1ª FERIA DE LA MICROINFORMATICA EN EL CORTE INGLES CASTELLANA

En la 6ª Planta de El Corte Inglés Castellana, le esperan 1.000 m² dedicados de lleno a la Microinformática. Con las principales marcas del mercado de los Microordenadores Profesionales y Microordenadores Familiares, áreas dedicadas a Complementos y Librería, actividades para colegiales... y todos los Servicios de una gran Feria. Además, un equipo humano, especializado en los diferentes Microordenadores, contestarán a todas sus consultas.
Y no olvide nuestras facilidades de compra: Financiación hasta 24 meses. Con el Servicio Post-Venta y la garantía de El Corte Inglés.

Venga a conocer las novedades de las mejores marcas, en la 1ª Feria de la Microinformática en El Corte Inglés.



TOTAL=(IMPORTE)—(IMPORTE) ×(DESCUENTO)/100

En la fig. 4, podemos ver el aspecto de uno de los registros de este fichero de facturas una vez que hemos introducido datos en él, mediante la opción ENTER.

Generación de listas

Para ilustrar los ejemplos que siguen, vamos a utilizar el fichero de datos bibliográficos que hemos creado. Para ello, hemos introducido los datos correspondientes a un conjunto de libros.

El **Superbase** ordena los registros de un fichero, en orden alfabético, según el contenido del campo clave del registro. Todo registro tiene que tener su campo clave, que define el usuario a la hora de diseñar el registro, teniendo en cuenta que los registros irán ordenados según el contenido de este campo.

En nuestro fichero de libros, hemos escogido el título del libro como campo clave. Esto quiere decir que, a medida que añadimos nuevos registros, el **Superbase** los ordena alfabéticamente según el título.

Vamos a ver como se pueden generar e imprimir listas de registros. Supongamos, en primer lugar, que queremos obtener una lista de los libros del registro, pertenecientes a la editorial **Bruguera.** Para ello, utilizamos la opción FIND que genera y almacena una lista con las claves de los registros seleccionados. Para generar la lista, escogemos la opción FIND desde el menú uno, pulsando F3, con lo que en la pantalla nos aparece un registro en blanco.

Ahora, para especificar el criterio de selección, esto es, que el contenido del campo editorial, basta con movernos, utilizando las teclas de cursor, hasta colocarnos en el campo de nombre Editorial, y escribir "**Bruguera**". Después, pulsando SHIFT+RETURN, se genera la lista.

Utilizando el comando OUTPUT, podemos pasar a imprimir los registros almacenados en la lista. Este comando permite una gran flexibilidad a la hora de imprimir. En primer lugar, aunque la lista sólo contiene las claves de los registros seleccionados, en nuestro caso las claves son los títulos de los libros, a la hora de imprimir, podemos utilizar información de cualquiera de los campos del registro, y no solo del campo clave. Nosotros hemos decidido que se impriman el título y la editorial, con lo que podemos comprobar que todos los registros de la lista corresponden a la editorial, tal y como habíamos especificado. Además, el comando OUTPUT permite imprimir cualquier texto o carácter entre los contenidos de los campos. Nosotros hemos escogido que se impriman 4 caracteres gráficos. Todas estas órdenes se introducen escribiendo una línea de comandos como esta:

print from "lista" (Título) "(caracteres gráficos)" (Editorial)

El resultado es el de la fig. 5 donde se observa que los libros van ordenados alfabéticamente según su título.

Como segundo ejemplo, vamos a obtener una lista, de todos los libros que incluyen en su título las palabras "micro", "computer" o "prog". Para ello, seleccionamos de nuevo la opción FIND y al aparecernos el registro en blanco, nos situamos sobre el campo del título y escribimos las tres palabras, separadas por (/), como se ve en la fig. 6. Al pulsar SHIFT+RETURN, se genera la lista, con las claves de los libros que cumplen con lo especificado. Para imprimir la lista, usamos de nuevo OUTPUT, pero ahora vamos a imprimir el título y el autor, en lugar del título y la edito-

```
mode : Select Match Data   :# 5   d

Titulo [micro/prog/computer          ]

Autor  [                             ]

Tema   [                             ]

Editorial [                          ]

Fecha de edicion  [        ]

Dimensiones [          ] No.pag.[    ]

Precio  [         ]
```

```
Computer data processing        ..... Davis
Introduction to computers       ..... Davis
Microprogramming primer         ..... Katzan
Nuev. tecnicas de programacion  .....
Structural programming          ..... Dijkstra
Systems programming             ..... Donovan
```

```
Graham Greene      El americano impasible
Henry James        Otra vuelta de tuerca
Henry Miller       Primavera negra
Isaac Asimov       Compre Jupiter
Isaac Asimov       Los propios dioses
```

rial, y además, en vez de caracteres gráficos, vamos a utilizar puntos para separar los dos campos. El resultado es la lista de la fig. 7, donde se puede observar, otra vez, que los libros van ordenados alfabéticamente según el t-ordenados alfabéticamente según el título.

Por último, vamos a ver el empleo de la función SORT en el manejo de estas listas. Hasta ahora, hemos visto que todas las listas generadas, van ordenadas alfabéticamente según el título del libro. Esto es así, porque hemos escogido el título como campo clave, y el **Superbase** siempre ordena los registros según el campo clave. Pues bien, utilizando la opción SORT, se pueden reordenar un conjunto o todos los registros del fichero, según el contenido de otros campos que no sean el campo clave.

Como ejemplo, vamos a tomar la lista de libros pertenecientes a la editorial, que obtuvimos anteriormente, y vamos a reordenarla según el autor. Para ello, y sabiendo que tenemos la lista almacenada con el nombre de "lista", basta escoger la opción SORT desde el menú 2, pulsando F4, y escribir en la línea de comandos:

from "lista" on (Autor) to "lista A"

Al pulsar RETURN, el **Superbase** toma los registros de "lista", los reordena según el contenido del campo autor, y los almacena en una nueva lista que hemos denominado "lista A".

Ahora, y para comprobar que la reordenación ha sido hecha correctamente, utilizamos la opción OUTPUT para imprimir "lista A", escogiendo en este caso título y autor, como campos a imprimir, en lugar de título y editorial. El resultado es el de la fig. 8 donde se observa como los libros van ordenados alfabéticamente según el autor.

Hemos revisado, con algunos ejemplos, tan solo una pequeña parte de las posibilidades de el **Superbase.** El conjunto de todas las posibilidades de actuación, es muy amplio, y queda fuera del alcance de este artículo, pero al menos, esperamos haber aclarado algunas ideas y conceptos, sobre esta Base de Datos, que puedan servir de orientación al lector interesado.

VALORES MAXIMOS DEL SISTEMA

Base de datos .	Nombre de la B.D.	16 caracteres
	N.° de B.D.	ilimitado
	N.° de ficheros en una B.D.	15
	Programas en una B.D.	ilimitados
	Listas de claves en B.D.	ilimitado
Fichero	Nombre de fichero	16 caracteres
	Registro por fichero	ilimitado
	Ficheros a la vez	1
	Tamaño de un programa	4K
Registro	N.° de campos	127
	Texto descriptivo en el diseño del fichero	2K
	Longitud del registro	1108 bytes
Campos	Nombre del campo	12 caracteres
	Texto	255 caracteres
	Clave	30 caracteres
	Numérico	9 cifras

TECNICAS DEL "SLOT"

Sistemas explotables por más de un microprocesador: Introducción a las técnicas del "SLOT" y Multisistemas

Desde el comienzo de la expansión de los sistemas de Microordenador, los proyectistas han intentado desarrollar métodos que permitiesen la explotación del *Software* de procedencia indiscriminada por un solo sistema y sin necesidad de alteraciones en el *hardware* del mismo, como un paso más hacia el soñado "ORDENADOR UNIVERSAL", capaz de simular el comportamiento de cualquier sistema existente en el mercado.

En la actualidad existen dos tendencias distintas que son las siguientes:

A) **Sistemas con *"Slots"*:** El nombre de *"Slot"* (ranura), procede de la existencia de un conector destinado a recibir una tarjeta equipada con un microprocesador (y un *hardware* asociado) que permite, por cambio de las tarjetas, seleccionar cualquier tipo de microprocesador con un coste muy bajo y poder así acceder al *Software* disponible para el mismo.

Estos sistemas se estructuran con un diagrama funcional como el indicado en la fig. 1.

Como puede verse en la misma, la tarjeta que se inserta en el *SLOT* incluye el programa de traducción de instrucciones entre ambos microprocesadores que residente en ROM es conectada directamente sobre el bus del sistema "A". El intercambio de los datos de proceso se realiza a través de 4 *buffers* de RAM, que funcionan unidireccionalmente explotados en DMA (acceso directo a memoria por ambos sistemas). Las señales de reloj R/W, etc., son generadas por el propio sistema, que también facilita la alimentación a la placa con el microprocesador "B". Se ha prescindido del resto de los elementos (I/O, controlador, etc.) para simplificar la representación gráfica del conjunto.

En la actualidad son varias las firmas de ordenadores que incluyen el *Slot* como equipamiento estándar de sus modelos, tal sucede con el **Commodore 720** entre otros, por ejemplo.

B) **Multisistemas:** El multisistema es una técnica sensiblemente distinta de la anterior, que se emplea cuando el número de microprocesadores distintos se restringe a 2 ó 3 como máximo, por buscarse la ambivalencia del *software* por motivos concretos (*software* de operación, *software* de comunicación, etc.).

Un buen ejemplo de este tipo de sistemas corresponde al sistema **CBM 9.000,** desarrollado por la **Universidad de Waterloo** en colaboración con **Commodore Bussines Machines.** Su sistema operativo se estructura como puede verse en la fig. 2.

Como puede observarse en la misma, ambos microprocesadores (el 6502 y el 6809) poseen sistemas operativos residentes en ROM independientes y comparten zonas de RAM (tanto los 32 K RAM del sistema original, como las 64 K del sistema

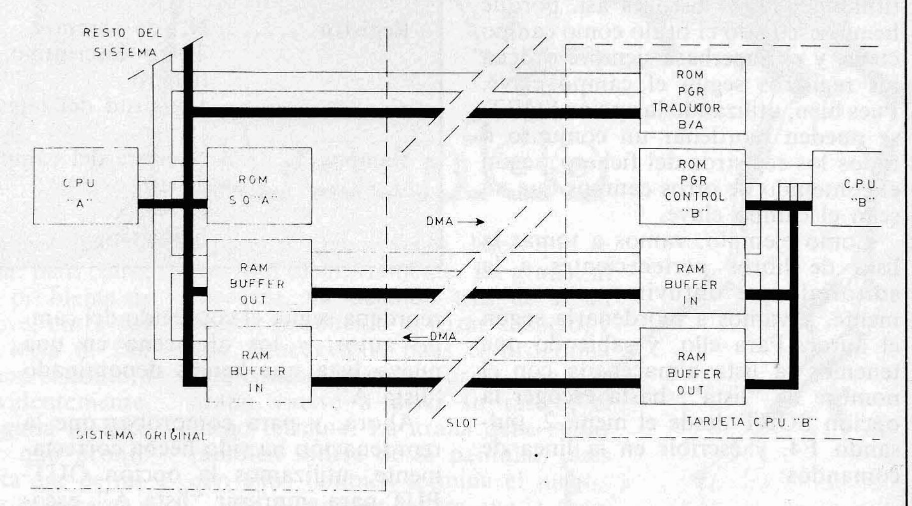


Figura 1. Sistema con Slot.

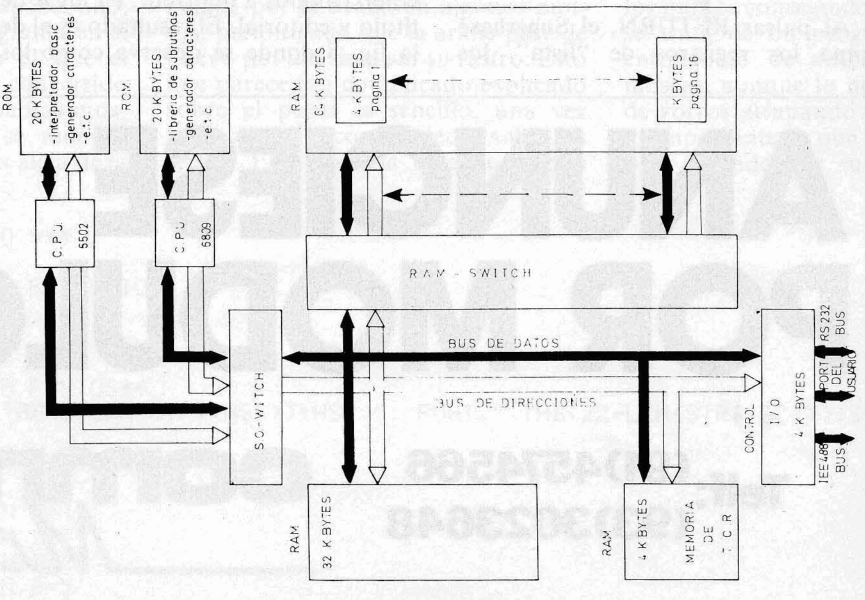


Figura 2. Multisistemas.

Y MULTISISTEMAS

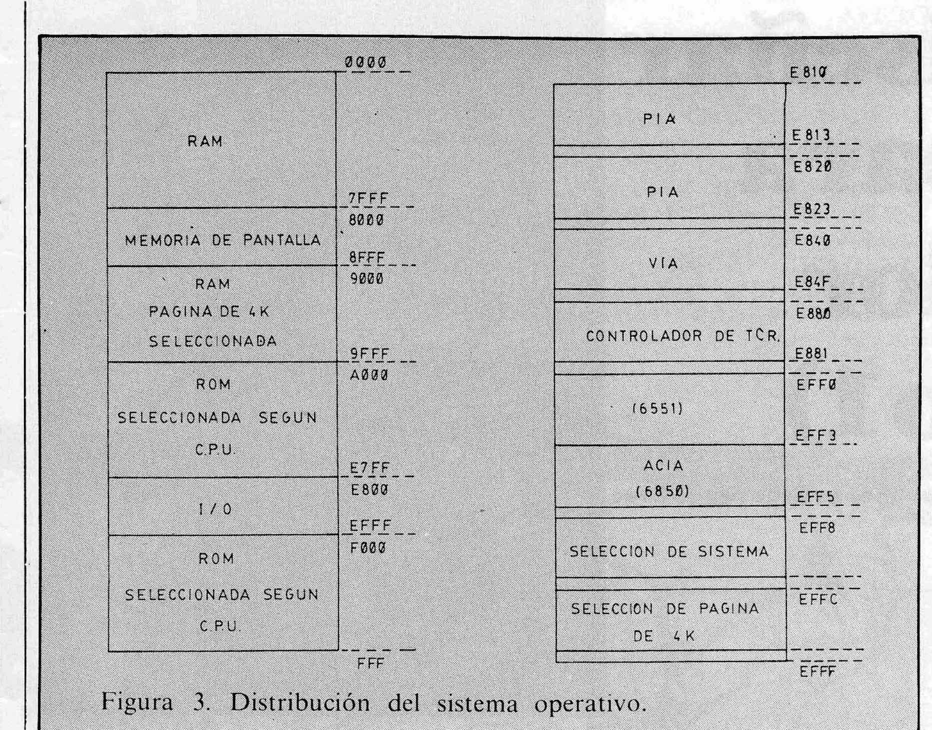


Figura 3. Distribución del sistema operativo.

Waterloo) y todo el dispositivo de control de periferia y I/O original de la serie **CBM,** potenciado con un *interface* bidireccional **RS232-C.**

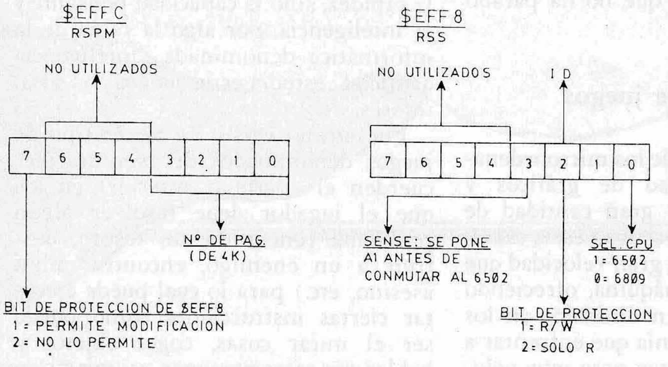
La conmutación del total del sistema compartido a uno u otro microprocesador puede realizarse por un *switch* en el sistema operativo. Otro *switch* permite seleccionar 1 de los 16 bloques de 4 K de los 64 K de RAM aportada por el sistema **Waterloo.** Esto permite explotar un total de 32 K + 64 K = 96 K RAM, lo que excede a la capacidad de direccionamiento de ambos microprocesadores (64 K).

Ambos microprocesadores pueden explotar el total de la RAM disponible. El 6502 con el S.O. original y con su Basic Residente y almacenamiento de variables, quedando disponibles los 32 K originales para el texto Basic. Igualmente, el 6809 puede explotar distintos compiladores e interpretadores que se cargan desde disco, en los 32 K originales. Estando disponibles en la actualidad interpretadores **COBOL, FORTRAM, BASIC D, APL, PASCAL** y ensambladores para el 6809; así como potentes subrutinas y programas de comunicación que permiten a la Unidad trabajar como terminal remoto **RS232-C** de cualquier sistema, siendo totalmente programables las características y la velocidad de comunicación desde 600 baudios hasta 9.600.

El total del sistema operativo se estructura como puede verse en la fig. 3.

Igualmente pueden ser seleccionados distintos *sets* de caracteres (incluidos los especiales APL), como se ve en la fig. 4-B y una separación adicional de un *pixel* puede ser programada para los mismos por control de las posiciones $E880 a $E8881 inclusive, cambiando la matriz de 8×8 por otra de 7×7 *pixels.* En resumen, las prestaciones del sistema original quedan extraordinariamente potenciadas, resultando muy adecuado tanto para aplicaciones docentes de comunicaciones e incluso industriales ya que a la mayor velocidad del 6809 los programas interpretados en éste son casi tan rápidos como los compilados en el 6502 lo que hace que sea adecuado a estos fines.

Joaquín Roca Dorda
Profesor Cátedra de Electrónica
E.U.P. Cartagena

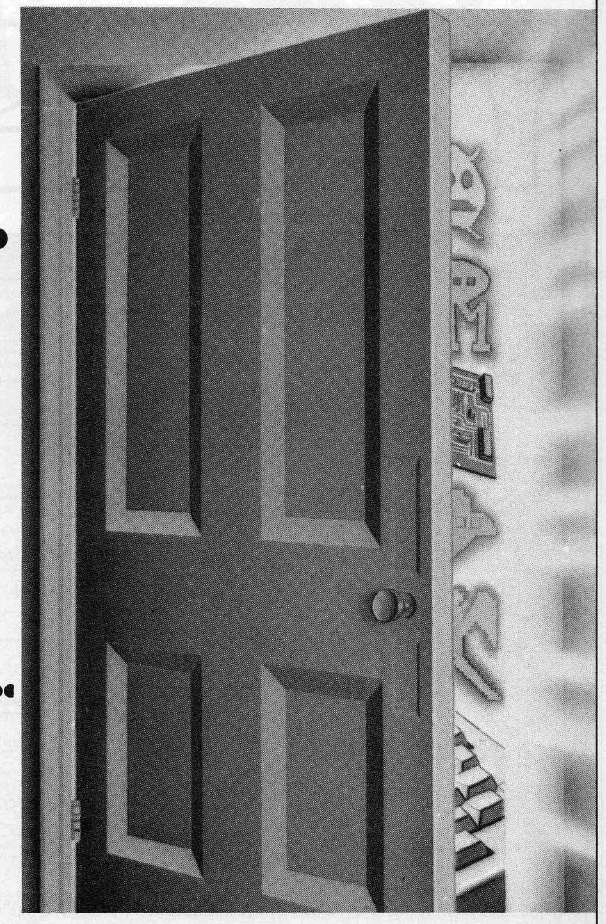


A. Estructura de los "switchs"
Figura 4. Otros controles.

S.O.	ØE88Ø	$E881	$E842	Nº CARC	TIPO
SET A	ØC	1Ø	ØC	255	MAYUS/MIN
SET B	ØC	1Ø	ØE	255	GRF/MAYUS
SET C	ØC	3Ø	ØC	255	MAYUS/MIN
SET D	ØC	3Ø	OE	255	APL

B. Selección de los distintos sets

Cómo diseñar juegos para ordenador (capitulo 1)



No vamos a empezar el artículo explicando que es un juego, cuestión innecesaria, pero si fijando algunas ideas sobre su implementación en los ordenadores y a partir de ese punto desarrollar las demás ideas.

Ya con la aparición de los primeros ordenadores comerciales (en los lejanos sesenta) muchos programadores empezaron a hacer programas como medio de diversión, debido a las limitaciones de las máquinas estos juegos eran bastante sencillos y no tenían gráficos, sonido o cualquiera de las características a las que estamos acostumbrados hoy en día. Entre estos primeros programas podemos destacar el **STAR TREK** (el primer juego, o casi) que todo ordenador que se preciase debía tener, asi mismo surgió un programa llamado **ADVENTURE** (aventura) y que dio nombre a un tipo de juegos que más adelante explicaremos.

Esos programas y los que le siguieron pusieron la piedra básica de los juegos por ordenador: utilizarlo como máquina pensante para que nos pusiese problemas y situaciones totalmente imprevisibles que había que resolver.

Con la llegada del microprocesador en la década de los setenta empezaron a surgir los ordenadores personales, ordenadores que cualquiera podía tener en su casa y que tenían características distintas a las de los *"mainframes"* (grandes ordenadores) muchos disponían de gráficas y sonido, trabajaban en tiempo real, es decir, siempre le hacían caso al usuario no como en las grandes máquinas que al tener que hacer caso a muchas personas solían dejar al usuario "colgado" a la espera de los datos que le había pedido. La aparición de los primeros juegos para ordenador personal tuvo un gran éxito que no ha parado hasta hoy día.

Distintos tipos de juegos

Con la aparición de los microordenadores con capacidad de gráficos y sonidos surgió una gran cantidad de juegos que se aprovechaban estas características unidas a la gran velocidad que proporcionaba la máquina, ofreciendo juegos con una gran dinámica en los que la persona se tenía que enfrentar a unos oponentes tontos pero muy veloces y en superioridad numérica, son los tipos de juego conocidos como marcianitos (en inglés se les llama ARCADE) y que podemos ver en cualquier bar, esta denominación es genérica aunque luego el programa no sea de marcianos, sino de bolas que comen puntos, tanques, o cualquiera de las múltiples variaciones sobre el tema.

En contraposición a estos juegos existen los juegos inteligentes, en estos se aprovecha la capacidad pensante de la máquina para ofrecer un oponente de la misma categoría que el ser humano, al que muchas veces sustituye (ajedrez, otello, etc.). Aquí ya no prima la rapidez, sino la capacidad pensante y la inteligencia, por algo la rama de la informática denominada "inteligencia artificial" estudia estos juegos con gran interés.

Por último existe un tercer tipo de juegos denominados de aventura (recuerden el apartado anterior) en los que el jugador debe resolver algún problema (encontrar un tesoro, destruir a un enemigo, encontrar a un asesino, etc.) para lo cual puede ejecutar ciertas instrucciones como puede ser el mirar cosas, coger objetos y hablar con otras personas, asi mismo se puede desplazar por los distintos lugares en que sucede la acción, que en la

mayoría de los casos no caben a la vez en la pantalla del televisor. Este tipo de juegos es el punto intermedio de los dos anteriores, la velocidad no es fundamental, aunque se puede hacer que sea en tiempo real y el jugador debe desarrollar más inteligencia que habilidad para poder ganar, porque estos juegos tienen final como los de "inteligencia, al contrario que los "arcade" en los que los enemigos son infinitos.

Naturalmente estas tres clasificaciones no son extrictas, pudiendo haber juegos que sean mezcla de estos tipos, o que no corresponda a ninguno (bastante raramente) y no por eso tienen que ser malos, al contrario las innovaciones siempre son buenas.

Vamos a hacer un juego

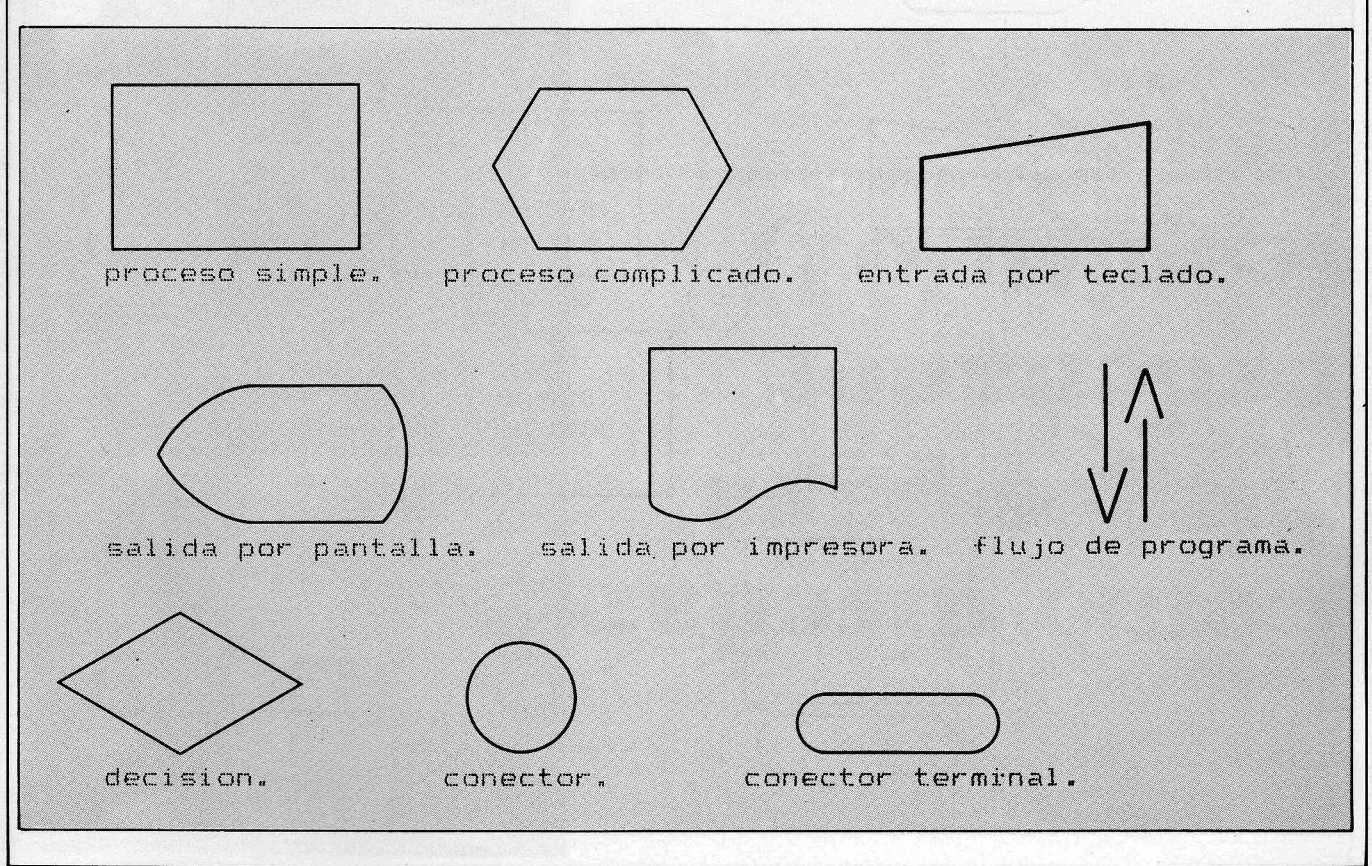
Una vez que hemos decidido hacer un juego conviene que nos sentemos tranquilamente con papel y lápiz y una buena dosis de café (el ordenador de momento conviene dejarlo guardado). Ante todo debe fijarse bien el objetivo a conseguir: ¿una partida de damas?, ¿los típicos marcianitos? o quizás una aventura en una isla solitaria imitando a **Robinson Crusoe**?. Una vez decidido esto debemos recordar que hacer un juego no es un juego, aunque esta frase parezca extraña, es real, una de las cosas más difíciles de programar es un juego. Mientras que en cualquier otro tipo de programas sabemos exactamente lo que hay que hacer y como se debe hacer, reduciéndose el trabajo a traducir esas instrucciones a un lenguaje comprensible por el ordenador; en un juego además de esto debemos: decidir cuál es el objetivo, de qué medios dispone el jugador para conseguirlo, como van a ser sus oponentes, crear la estrategia de estos, como se van a mostrar los datos por pantalla y un largo etcétera de problemas que en otro sitio no se nos presentarían. Una vez definidas y resueltas estas cuestiones debemos pasar a la siguiente fase de diseño.

Construcción de un organigrama

Aunque bastante odiado por muchos programadores (incluso profesionales) la realización de un organigrama antes de empezar a escribir el programa es sumamente útil porque permite un diseño más racional además de permitirnos descubrir los problemas sobre el papel, siendo más fácil su corrección que si estamos a la mitad del programa cuando lo estamos tecleando "por improvisción".

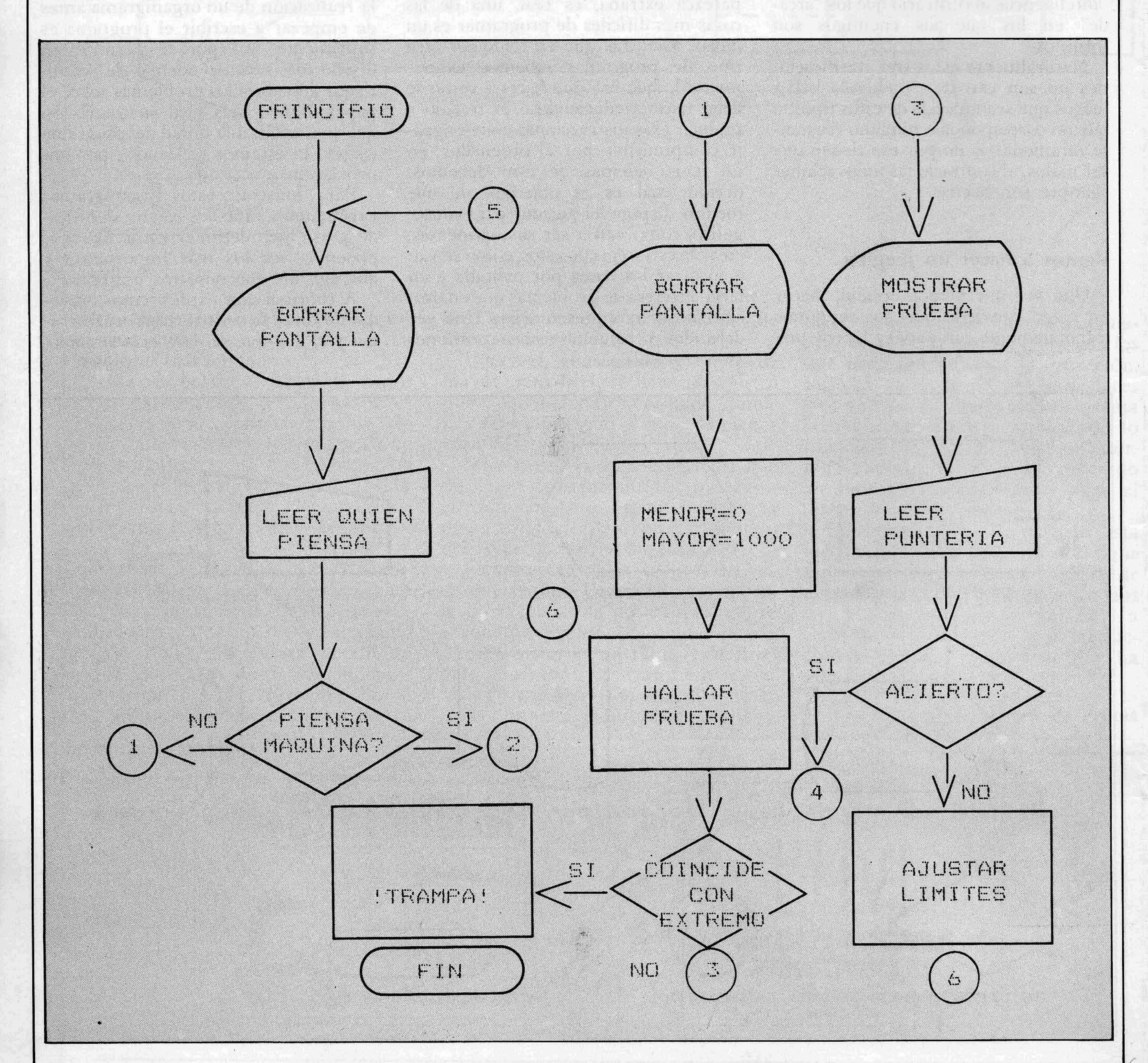
Para construir estos organigramas existen unos símbolos con un significado y uso bien definido. En la figura 1, podemos ver los más importantes y que son los que nosotros usaremos.

A continuación explicaremos brevemente como se construye un organigrama y el uso que se da a cada símbolo.



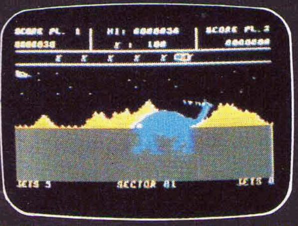
Un organigrama es el conjunto de las instrucciones del programa explicadas en forma de bloques, no se explica el funcionamiento instrucción por instrucción sino es paquetes de instrucciones de fácil programación, así un bloque podría ser: MUEVE FICHA HASTA LA CASILLA ELEGIDA. La traducción de este bloque a instrucciones de programa será, evidentemente, más larga aunque no revierta una excesiva complicación, incluso se podría diseñar una subrutina general que se usase cada vez que tuviésemos que hacer esta operación Los bloques están unidos por líneas de flujo que salen de uno y van a parar a otro, pudiendo llegar solo a uno, ya que los programas son lineales (no se pueden seguir dos líneas a la vez).

Al principio y al final del programa se deben colocar conectores terminales con la palabra PRINCIPIO o FIN escrita dentro de ellos, su finalidad precisamente consiste en indicar esto; donde empieza y termina el programa indicación que pese a parecer superflua

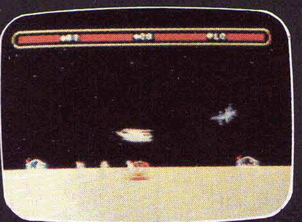


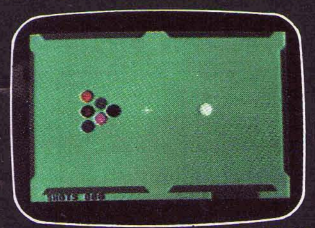
indescomp
SOFTWARE
COMMODORE 64

UNA NUEVA GENERACION DE VIDEOJUEGOS


MUTANT CAMELS Cass. 2.200

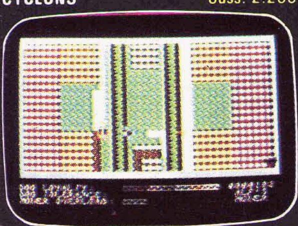

MATRIX Cass. 2.000


CHOPFLITER Cart. 8.170


HUSTLER Cass. 2.200


REVENGE MUTANT CAMELS Cas. 2.200


CYCLONS Cass. 2.200


HOVER BOVER Cass. 2.200

COMMODORE 64

SIREN CITY Cass. 1.900

CUDDLY CUBURT Cass. 1.900

KONG Cass. 2.000

SKRAMBLE Cass. 1.700

MOON BUGGY Cas. 2.000

COMMODORE 64

- SPRITE MAN 1.900
- FROGGER 64 1.700
- SNAKE 1.800
- ANIHILATOR 1.600
- PAKACUDA 1.700
- EXTERMINATOR 1.900
- VORTEX RAIDER 1.700
- DEFENDER 64 1.700
- LASER ZONE 1.700
- CHOPFLITER 8.170
- ZAXXON 7.190
- JUMPMAN 7.190
- FORT APOCALIPSE 6.290
- FUTBOLIN 1.800
- HORACIO 2.200
- PROCESADOR TEXTOS 3.300
- MAILING Y ETIQUETAS 3.300
- CONTABILIDAD PERSONAL 3.300
- BASE DE DATOS FICHERO 3.300
- ULTRABASIC 2.900
- SINTHY 64 2.900
- FORTH TINY 2.900

SINCLAIR SPECTRUM*

- PENETRATOR
- TIME GATE
- EL JUGADOR DE AJEDREZ
- ASTROBLASTER
- COMECOCOS
- HORMIGAS
- TRAXX
- DISEÑADOR DE JUEGOS
- LA PULGA (BUGABOO)
- BILLAR AMERICANO
- EL PINTOR
- MISILES
- RAPTORES DE LA GALAXIA
- WINGED AVENGER
- INVASORES & GALAXIANS
- FORTH
- BASE DE DATOS
- MATEMATICAS
- ARITMETICAS BASICAS
- TUTOR

APPLE 900 PROG.

ATARI 800 PROG.

SINCLAIR ZX81 100 PROGRAMAS

DRAGON 32 60 PROGRAMAS

VIC 20 200 PROG.

IBM 350 PROG.

SOLICITE GRATIS CATALOGO GENERAL INDICANDO EL ORDENADOR SOBRE EL QUE LO DESEA


BUGABOO LA PULGA Cassette 2.000

Ya a la venta en


El Corte Inglés

Sonitel

CHIPS & TIPS

Distribuidores autorizados y por Correo

ENVIAR A: indescomp, Castellana, 179. Tel. 6563012. Madrid-16

CANTIDAD	PRODUCTO	PRECIO UNIT.	TOTAL
EMPAQUETADO Y TRANSPORTE	GASTOS DE ENVIO	300	
		TOTAL	

Incluyo cheque nominativo a favor de INDESCOPM, S.A. por .... Pts.

Remitan el pedido contra reembolso a:
D. .... Dirección ....
n.° .... Provincia .... Tel. .... Profesión ....

* Los programas para Spectrum son comercializados únicamente y en exclusiva por Investronica, S. A.

Estamos en la Feria de la Informática.
"EL CORTE INGLES" Castellana, del 4 al 28 de Abril.

a veces no lo es tanto (además crea un buen efecto). Entre estos dos conectores debe hallarse el resto del programa, para unir las distintas partes entre si se utilizan las líneas de flujo; que indican cual es el siguiente bloque que se ejecuta después de este. Los procesos simples son aquellos conjuntos de instrucciones que no hay que especificar más ya que su transformación en líneas de programa es inmediata, el bloque ENCONTRAR EL MAYOR ELEMENTO DE LA MATRIZ A(10) se transforma el las líneas del programa corto de este artículo.

En cambio las operaciones que se designan como proceso complicado requieren una explicación aparte de su funcionamiento, un caso típico sería: HALLAR LA MEJOR CASILLA POSIBLE PARA MOVER. Esta operación en muchos casos es difícil de plantear y de programar y se debe especificar bien como se hace.

La entrada por teclado significa exactamente eso, el programa lee datos del teclado para darles un uso explicado posteriormente (mover el marciano, leer la jugada del jugador humano en una partida de damas, recibir órdenes para nuestro explorador...), asi mismo la salida por pantalla y/o la salida por impresora dan resultados (números, dibujos o de cualquier otro tipo) por el medio que especifiquen.

El símbolo de toma de decisión es el único del que salen dos líneas de flujo, aunque esto parezca contradecir el postulado que dimos antes de que solo parezca contradecir el postulado que dimos antes de que solo puede haber una línea de flujo ejecutándose a la vez, no es así, este símbolo representa una toma de decisión en base a unos datos y según el resultado se sigue una línea o la otra. Pero nunca las dos a la vez.

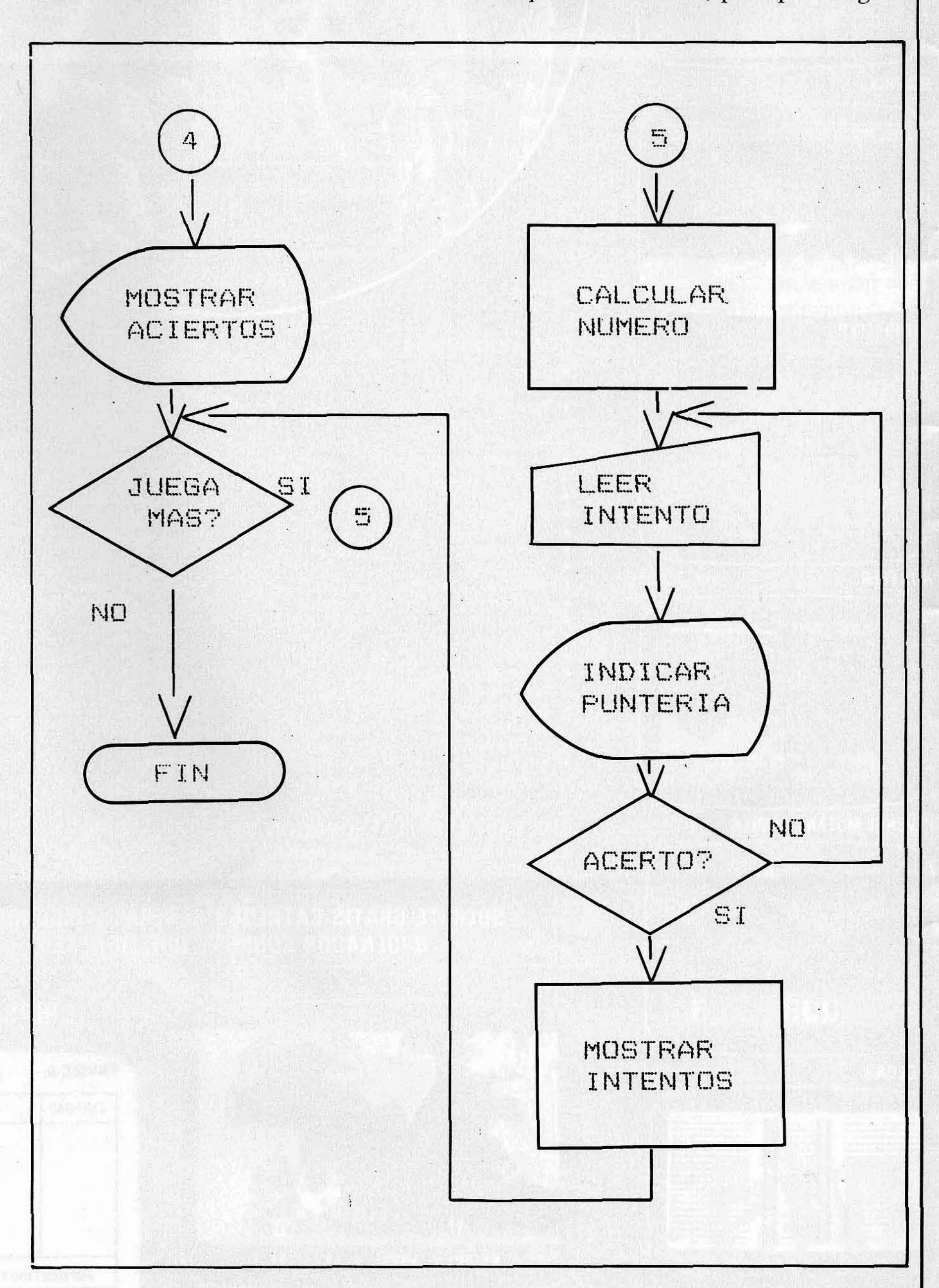
Por último el conector (No terminal) es de gran uso cuando se escriben programas largos que no caben en una sola hoja, en este caso al final de la línea de flujo se pone este conector con un símbolo dentro, en la página donde continua la ejecución ponemos este conector con el mismo símbolo seguido del resto del proceso, por decirlo de algún modo es un indicador de donde continua el programa.

Introducción del programa

Ya ha llegado el momento de introducir el programa, podemos desempolvar el ordenador y enchufarlo porque vamos a trabajar con el. En este punto debemos traducir las instrucciones del organigrama a instrucciones comprensibles por la máquina, si hemos hecho el organigrama bien esto no representará mucha dificultad y será sólo una cuestión de paciencia.

Eliminación de errores

El apartado anterior no es el último, aunque debería serlo, pero por desgra-



cia los duendes del silicio andan sueltos y un programa que debería funcionar, no funciona o funciona mal. En vez de tirarlo a la basura (opción que atrae) lo que se debe hacer es empezar a repasar el organigrama y el programa de ordenador para ver donde falla y porqué. Este proceso no es sencillo (muchas veces es el más complicado) y se debe realizar con mucho cuidado y paciencia porque muchas veces el error es un cero que debería ser la letra "o" o un "1" que debería ser un cero. Una vez hecho esto a disfrutar del juego que para eso lo hemos hecho.

Estos consejos que se dan a continuación son muy importantes para que el programa sea divertido y nos agrade.

Ante todo debemos recordar que hay que luchar contra un enemigo imaginario y no contra un ordenador que intenta hacer todo lo posible por evitar que juguemos, las instrucciones deben ser claras, el manejo sencillo (evitar que para indicarle que queremos ir a la izquierda tengamos que pulsar 4 teclas, con una debe bastar), no debe ser demasiado difícil como para frustrar al principiante ni demasiado fácil como para aburrir al experimentado. Un buen sistema para evitar esto último es la dificultad progresiva, cuanto más avanzamos en el programa más difícil se vuelve. El jugador humano puede ser atacado, un juego en el que nunca podemos tener algún método de defensa para poder permitirnos la satisfacción de destruir al malvado enemigo cibernético (naturalmente usando medios pacíficos, no clavando un hacha en medio del teclado), por último a los juegos de acción, sería aconsejable dotarles de una tecla que parará momentaneamente la ejecución del juego (no hay nada más horrible que una llamada telefónica en medio de una partida de comecocos) pudiéndose continuar al volver a pulsar la tecla.

Un ejemplo sencillo

Como ejemplo de las ideas que acabamos de explicar vamos a realizar un sencillo programa.

Idea: El juego consiste en que uno de los dos contendientes (el ordenador o la persona) piense un número entero (sin decimales) entre el 1 y el 999 y el otro lo debe de averiguar poniendo números de prueba y siguiendo las indicaciones del contrario que le debe indicar si el número es más alto o más

```
5 REM MA= MAYOR ELEMENTO, OR= SU ORDEN
10 MA=A(1):OR=1
20 FOR I=2 TO 10
30 IF A(I)>MA THEN MA=A(I):OR=I
40 NEXT I
```

```
10 PRINT "♥"
20 INPUT "PIENSO YO?";R$
30 IF LEFT$(R$,1)="S" THEN GOTO 100
40 IF LEFT$(R$,1)="N" THEN GOTO 500
50 GOTO 10
100 NU=INT(RND(0)*999)+1
110 IN=0
120 PRINT "♥"
130 INPUT"NUMERO A PROBAR:";PR
140 IN=IN+1
150 IF PR<NU THEN PRINT "EL NUMERO ";PR;" ES DEMASIADO BAJO":GOTO 130
160 IF PR>NU THEN PRINT "EL NUMERO ";PR;" ES DEMASIADO ALTO":GOTO 130
170 PRINT "ACERTO EL NUMERO EN ";IN;"INTENTOS"
180 GOTO 200
200 INPUT "DESEA JUGAR OTRA VEZ?";R$
210 IF LEFT$(R$,1)="S"THEN GOTO 10
220 IF LEFT$(R$,1)="N"THEN END
230 GOTO 200
500 PRINT "♥"
505 IN=0
510 ME=0
520 MA=1000
530 PR=INT((MA-ME)/2)+ME
535 IF (PR=ME) OR (PR=MA)THEN PRINT "ME HAS MENTIDO, YA NO JUEGO":END
540 IN=IN+1
550 PRINT "ES EL ";PRU;"?"
560 INPUT "ALTO, BAJO O CORRECTO";R$
570 IF LEFT$(R$,1)="C" THEN PRINT "LO ACERTE EN ";IN;" INTENTOS.":GOTO 200
580 IF LEFT$(R$,1)="B" THEN ME=PR:GOTO 530
590 IF LEFT$(R$,1)="A" THEN MA=PR:GOTO 530
600 GOTO 550
```

bajo que el que ha pensado. La estrategia que debe seguir el ordenador es la siguiente: Si le toca pensar a él debe generar un número entre el 1 y el 999 aleatoriamente a continuación debe empezar a "leer" los datos que le dé la persona y decir si son mayores o menores que el que ha pensado, si coinciden indicará que lo acertó. Si le toca adivinar el número a él, lo hará siguiendo el proceso denominado de búsqueda dicotómica, este proceso aunque largo para determinados números tiene una efectividad media bastante buena. Básicamente consiste en coger el intervalo en el que está comprendido el número, sabiendo que este no es ninguno de los dos extremos, por lo tanto de partida cogeremos los números 0 y 1000, que son ilegales y entre los que debe estar comprendido el número a adivinar, el número a probar es el que se encuentra en la mitad del intervalo (en el primer intento será el 500), si no es este número se le cogerá como nuevo extremo inferior o superior dependiendo de si es demasiado alto o demasiado bajo y se vuelve a repetir el proceso hasta que se encuentre el número, por ejemplo, si el número pensado es el 700 en el primer intento nos preguntará que si es el 500, al decirle que es demasiado bajo lo tomará como límite inferior quedando el número buscado dentro del intervalo (500, 1000). Volverá a calcularse el número de prueba y a preguntar (el 750), ajustando los límites otra vez, etcétera.

Además debe indicar el número de intentos que se tarda en averiguarlo.

Una vez descrito el programa pasamos a escribir el organigrama que es el de la figura 2.

Además, cada vez que se lea o se escriba una prueba hay que incrementar el contador de intentos en una unidad, este contador al principio se habrá puesto a cero. Estos procesos no se han incluido en el organigrama por no llenarlo excesivamente y son de fácil comprensión.

Después del organigrama escribiríamos el programa correspondiente y una vez funcionando lo dejaríamos "bonito" resultando el listado de la figura 3.

Un atento estudio (hágalo como problema de estudio) le revelará la coincidencia del programa con el organigrama, aunque esta semejanza en ningún programa llegará a ser del cien por cien debido a los problemas de última hora (a menos que hagamos el organigrama después del programa, práctica muy bonita pero sin utilidad aparente).

Por Fernando García Fernández

PROGRAMA: BEWITCHED
TIPO: JUEGO
DISTRIBUIDOR: ABC ANALOG
FORMATO: CASSETTE
COMPUTADOR: VIC 20

Cuando hoy se le da tanta importancia a la memoria del ordenador y se la considera imprescindible para cualquier cosa vemos un juego que funciona con la memoria mínima del *VIC* (8 Kbytes) y que además es divertido y atrayente, esperemos que esto dé de pensar a mucha gente.

Las instrucciones que vienen con la cinta son claras, aunque en inglés, lo que puede dificultar su manejo a algunas personas. Una vez cargada y puesta en funcionamiento empezamos a jugar. El tablero de juego es un inmenso laberinto cerrado de tramo en tramo por cuatro puertas, siendo cada una de un color, sólo una nos dejará pasar a la siguiente zona, mientras detrás de las otras tres hay una pared de piedra. En el nivel superior de la pantalla hay cuatro llaves fijas con cuatro colores distintos (uno por cada puerta), nosotros controlamos una quinta llave, negra al principio, que al pasar por una de las anteriores se cambiará de color y podrá abrir la puerta de su mismo color. Esta sencilla tarea se ve ligeramente "complicada" por la presencia de unos fantasmas que quieren comernos (¿qué otra cosa iban a querer hacer?), naturalmente las llaves están lo más alejadas posibles de la puerta correspondiente. Los primeros niveles son sencillos, pero una vez que nos vamos adentrando en el laberinto el número de marcianos va aumentando y además si detrás de una puerta hay pared, tendremos que subir a cambiar de color al origen del laberinto (varios niveles más arriba) con lo que deberemos retroceder todo lo avanzado y volverlo a recorrer después.

Como se ve, la mecánica del juego es muy simple y no presenta complicación, lo que no implica que no sea divertido, al revés; es muy divertido yposee un alto grado de adicción. La calidad de las gráficas es muy buena para esta máquina, y no necesita ninguna expansión de memoria (es más, debemos desconectarla antes de apagarla porque si no no funciona). El control del juego se puede realizar con el teclado o con un *joystick* conectado al *port* de control.

■ ■ ■ □ □

PROGRAMA: SPEED/BINGO MATH
TIPO: EDUCATIVO
DISTRIBUIDOR: MICROELECTRONICA Y CONTROL
FORMATO: CARTUCHO
COMPUTADOR: COMMODORE 64

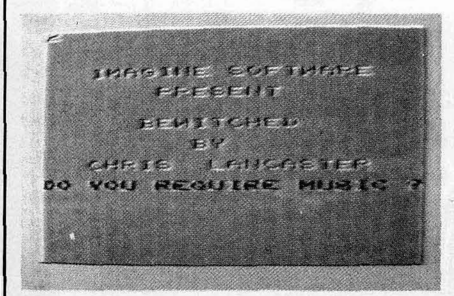
Este cartucho contiene dos juegos matemáticos básicos para que los niños pequeños, que están aprendiendo a sumar (y sus padres si lo necesitan) puedan aprender más rápidamente con un rato de diversión asegurada. Los dos programas combinan la acción de un juego de marcianitos con la rapidez mental de ls operaciones matemáticas básicas (suma, resta, división y multiplicación).

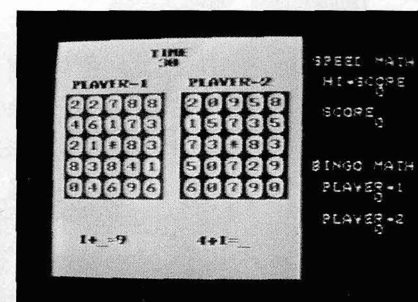
Para poner en marcha el programa basta conectarlo en la ranura dispuesta a tal efecto en la parte trasera de la máquina, habiendo tomado antes la precaución de apagarlo. Una vez hecho esto enciéndalo y le aparecerá una pantalla con las instrucciones (todo en inglés) preguntándole cuál de los dos juegos prefiere. El *SPEED MATH* es una serie de 30 operaciones del tipo elegido por nosotros (sumas, restas, etc.) que debemos resolver en un tiempo máximo de diez segundos, dándonos una puntuación que depende del tiempo que hemos tardado en resolverla (evidentemente, si a los diez segundos no se ha resuelto, la puntuación será cero), no es necesario pulsar RETURN para indicar que se ha pulsado la respuesta correcta, esto es una buena idea ya que a muchos niños les cuesta aprender precisamente ese punto: que tienen que pulsar RETURN después de dar la respuesta correcta, de este modo no se marean y no tienen que aprender una cosa totalmente ajena al juego.

El BINGO MATH es un divertido juego para dos personas, en el a cada jugador se le plantean una serie de cálculos y debe resolverlos y, a continuación, llevar el cursor (un asterisco) a la casilla del cartón de "bingo" donde se halle la respuesta correcta, naturalmente todo esto lo debe hacer durante un tiempo máximo de cinco segundos y debe ser más rápido que su compañero, el primero de los dos que logre cinco respuestas en una línea hace bingo.

Aunque parezcan unos juegos limitados para cortas edades poseen un interés bastante elevado y muchas personas "mayores" deberían jugar para comprobar que no saben operar tan bien como se creen. La presentación está bien hecha y cumple perfectamente con su cometido didáctico.

■ ■ ■ □ □



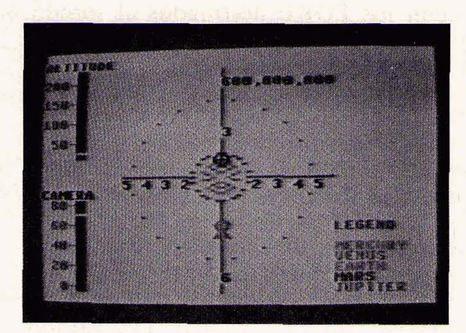


PROGRAMA: VISIBLE SOLAR SYSTEM
TIPO: EDUCATIVO
DISTRIBUIDOR: MICROELECTRONICA Y CONTROL
FORMATO: CARTUCHO
COMPUTADOR: COMMODORE 64

Hacer un programa educativo y que a la vez proporcione diversión no

es una tarea fácil, pero en este programa que visualiza el sistema solar la solución está muy bien conseguida, proporcionando un rato agradable a la vez que una información básica sobre el planeta en que vivimos y los que nos rodean.

Para ponerlo en marcha basta introducir el cartucho en el conector situado en la parte trasera del ordenador (¡apáguelo antes de hacerlo!) y encender la máquina. Automáticamente nos aparecerá una bonita pantalla de presentación con cuatro planetas girando alrededor del sol, seguida de una perspectiva cercana de la Tierra con la Luna girando alrededor de ella. Siguiendo las instrucciones del manual (en inglés) nos situamos en el interior de una nave espacial, con la que podremos recorrer el sistema solar y sus principales planetas (sólo seis de ellos, ¿hubiera sido tan difícil colocarlos a todos?) podremos verlo desde distintas perspectivas con la particularidad de que no es estático, los planetas se mueven en sus trayectorias mientras los vemos consiguiendo un muy bonito efecto. Asimismo, podemos ver una toma cercana de cada planeta, con este y sus satélites, girando y mostrándonos su superficie (por lo menos los principales detalles). Todos



estos pasos van acompañados en el manual de descripciones y datos de gran interés, como puede ser la composición de la atmósfera, su temperatura, etc.

Existe una subrutina denominada "ASTROCALC", que nos indicará los principales datos de cada planeta y nos permitirá compararlos con los de otro, viendo sus diferencias fundamentales, su estudio resulta francamente entretenido. Hay que indicar que todas las medidas se dan en el sistema americano (millas, grados Farenheit, el peso en libras...).

Educativamente está muy bien hecho y no aburre, aunque (como a todos los programas) se le podrían haber añadido un par de cosas más que lo hubieran mejorado mucho.

■ ■ ◪ □ □

Traducción del VIC al 64

Son bastantes las llamadas y cartas que hemos recibido, para saber si los programas escritos para el **VIC-20** pueden correr en el **64**. La respuesta la ofrece este artículo.

Por el contrario, ejecutar programas escritos para el **64** en el **VIC** es tarea menos abordable. Como punto de partida tengamos en cuenta que el **CBM 64** dispone de 64 Kbytes, bastantes más que el otro modelo, aunque se le añada la expansión de 16 K.

Aunque posteriormente ambas máquinas poseen un aspecto similar, y el **BASIC** es prácticamente idéntico, existen apreciables diferencias. Partiendo de los listados de programas desarrollados para el **VIC**, se pueden hacer correr en el **64**, simplemente copiando y haciendo las modificaciones pertinentes sobre la marcha.

Si nuestra intención fuera por el camino de intentar cargar en el **64** un programa previamente grabado en cinta por el **VIC**, lo más seguro es que el ordenador no entienda nada. Entre otras cosas, porque ambos utilizan diferentes velocidades para grabar y cargar programas con cinta.

Lo que vamos a ver es como efectuar cambios necesarios, sea un programa que se acaba de teclear y ya está en la memoria del **64**, o sea un listado del programa sobre papel.

Tratándose de un programa escrito en **BASIC**, las dos mayores diferencias se centran en la visualización en pantalla y los PEEK y POKE.

Las sentencias con PRINT no presentan dificultades sustanciales. En principio no haría falta alterarlas. Sin embargo, la posibilidad de los 40 caracteres por línea puede tener sus ventajas.

Los programas que tienen acceso a la memoria destinada contenido de pantalla son más delicados de convertir. En muchos se utilizan variables destinadas a manejar la visualización. En tales casos, la solución pasa por cambiar las variables clave, encargándose el programa del resto. Por ejemplo:

10 SC = 7680: PL = 23: LL = 32

En este caso sólo se cambian el puntero de pantalla, SC, y las variables de longitud de página y línea, PL y LL. Si el programa no utiliza esta técnica, entonces no existe alternativa, que no sea cambiar uno a uno todos los valores.

Encontramos un problema similar con los POKE destinados al sonido. En el **VIC** los cuatro registros de sonido se localizan entre las direcciones 36874 a 36877, estando el de volumen en la 36878. En registro del volumen se encuentra en la dirección 54296, para el **64**, pero no existe correlación entre los registros de uno y otro. Cada voz tiene cinco direcciones que la controlan, destinadas cada una al *attack/decay, sustain/release,* forma de onda y una para las frecuencias altas y bajas. Los valores de las notas son también diferentes.

Lo que puede resultar más sorprendente es que los programas en código máquina puedan resultar más fáciles de convertir. Esto se debe en primer lugar a que tanto el **VIC** como el **64** poseen un sistema operativo común, llamado **Kernal.** El **Kernal** es fundamentalmente una tabla de saltos, que puede ser llamada desde los programas del usuario.

Veamos, por ejemplo, que para obtener un carácter pulsado en el teclado, se puede llamar a una rutina bautizada como SCNKEY. Similarmente, para que aparezca un carácter, se llamará a otra por nombre CHROUT.

Estas rutinas son las mismas para ambas máquinas, de tal manera que si el programa utiliza el **Kernal** con profusión, serán necesarios pocos cambios. Por el contrario, si el programador utiliza sus propias rutinas para ejecutar tareas similares, será necesario reescribir toda la cofificación del programa.

Los ordenadores se comunican con el mundo externo por medio de los *ports,* que son los lugares físicos donde se depositan y recogen bits. Para que el ordenador se entienda con los periféricos u otros dispositivos (otros ordenadores incluidos), se necesita que haya protocolo común para ambos, esto es *software,* y por otro lado, el *interface* debe ser también del mismo tipo ésta es la parte *hardware. Interface* es la circuitería que tiene dos caras: la que da al dispositivo y la que da al exterior. Los estándares responden a determinados acuerdos sobre la disposición de las patillas el tipo de señales que acoge y los niveles de tensión, o corriente, que se relacionan con ellas.

Como es lógico, los ordenadores de **Commodore** también disponen de varios tipos de *interface* o, en su caso, existen módulos conectables, que hacen la conversión a partir de las señales producidas en los buses del sistema.

DIFERENCIAS ENTRE LAS LOCALIZACIONES DE MEMORIA

Principio del BASIC	4096	4608	2048
Memoria de pantalla	7680-8191	4096-4607	1024-2023
Memoria del color	38400-38905	37888-38399	55296-5695
Volumen	36878	36878	54296
Registros del sonido	36874-7	368-7	ver manual
Buffer del cassette	828-1019	828-1019	828-1019
Color de la pantalla	36879	36879	53281
Color del marco	36879	36879	53280

Esta no es una guía completa de las localizaciones de memoria de ambas máquinas. Simplemente muestra alguna de las diferencias importantes. En **VIC** expandido se hace referencia a los módulos de expansión de 8 ó 16 K. Con la de 3 K aparece otra diferencia.

SU PROGRAMA PARA CUALQUIER SISTEMA COMMODORE PUEDE HACERLE GANAR 5.000 PTAS.

EL PRESENTE CONCURSO ESTA ABIERTO A TODOS NUESTROS LECTORES Y SU PARTICIPACION E INSCRIPCION ES GRATUITA. LEA LAS BASES DEL CONCURSO

■ NO SE ESTABLECEN LIMITACIONES EN CUANTO A EXTENSION, TEMA ELEGIDO O MODELO DE ORDENADOR

■ LOS CONCURSANTES DEBERAN ENVIARNOS A LA DIRECCION QUE FIGURA AL PIE, EL CASSETTE O DISKETTE CONTENIENDO EL PROGRAMA, UNA EXPLICACION DEL MISMO Y, AL SER POSIBLE, UN LISTADO EN PAPEL DE IMPRESORA. SE PODRAN ENVIAR TANTOS PROGRAMAS COMO SE DESEE

■ LOS PROGRAMAS, PREVIA SELECCION, SERAN PUBLICADOS EN LA REVISTA, OBTENIENDO TODOS ELLOS 5.000 PTAS.

■ LA DECISION SOBRE LA PUBLICACION O NO DE UN PROGRAMA CORRESPONDE UNICAMENTE AL JURADO NOMBRADO AL EFECTO POR "COMMODORE MAGAZINE", SIENDO SU FALLO INAPELABLE

■ LOS CRITERIOS DE SELECCION SE BASARAN EN LA CREATIVIDAD DEL TEMA ELEGIDO Y LA ORIGINALIDAD Y/O SENCILLEZ EN EL METODO DE PROGRAMACION GLOBAL

■ ENVIAR A:
CONCURSO COMMODORE MAGAZINE
C/JEREZ, 3, MADRID-16



commodore Magazine

C/Jerez, 3 - Madrid-16

Iniciación al lenguaje má

En este capítulo vamos a empezar abordando una descripción más detallada de los demás bloques que componen este sistema microprocesador mínimo (fig. 1).

Como se puede apreciar en la figura, se diferencian cuatro grandes bloques:

— Memoria de programa
— Memoria de datos
— Módulos de entrada y salida
— Periféricos del sistema.

Todos estos bloques están interconectados por los tres *buses* del sistema. Dijimos anteriormente que un *bus* es un conjunto de líneas eléctricas (cables eléctricos), que conexionan las distintas partes del ordenador y por donde circular las señales bajo las cuales actúan los diferentes circuitos.

Bus de direcciones

El *bus* de direcciones está compuespor 16 líneas unidireccionales (llevan la información sólo en una dirección), distribuyéndose a todos los dispositivos que forman el microordenador. Al decir todos los dispositivos, también nos referimos a los dispositivos de *interface,* que son considerados por el microprocesador como posiciones de memoria.

¿Qué es lo que queremos conseguir con estas 16 líneas (llamadas *bus* de direcciones)? Se trata de que en cada acceso que realicemos, ya sea a una posición de memoria cualquiera o algún *interface,* sólo se encuentre habilitado un dispositivo simultaneamente. Este *bus* está preparado, por ejemplo, para excitar un dispositivo TTL (Transistor-Transistor-Logic) standard, pero en cualquier caso, mediante *buffers* (amplificador de señal digital) podríamos excitar varias cargas TTL.

Bus de datos

También conviene recordar que el *bus* de datos está compuesto por 8 líneas o caminos de señal, pero esta vez bidireccionales, es decir, estas líneas son capaces de llevar los datos en una u otra dirección, haciendo de vehículo de comunicación entre las diversas partes del sistema.

Otro tipo de *buffers* que controlan estas líneas, son triestado. ¿Qué significa ésto? pues nada más sencillo que tres estados, es decir; 1, 0, y un tercer estado capaz de poner las líneas en alta impedancia (desconexión) con sólo aplicarle una señal de control *(tri-estate).* El efecto producido, equivaldría a que desconectáramos físicamente las líneas del dispositivo al que no vayan dirigidas dichas señales. La lógica de tres estados es por tanto necesaria, por el hecho de que las líneas son bidireccionales.

La memoria ROM (Real Only Memory)

Conviene recordar que una memoria ROM, es una memoria de sólo lectura y cuyo contenido es permanente una vez grabada. Estas memorias son las idóneas para contener un programa fijo que controla el sistema.

La fabricación de las memorias ROM se realiza mediante micromáscaras y su programa es inmodificable por el usuario.

La memoria ROM contiene el programa MONITOR, que es el programa encargado del funcionamiento de todo el conjunto, incluyendo a los periféricos de que disponga nuestro sistema.

Además del programa Monitor, pueden existir otros programas en memoria ROM, para facilitar el trabajo de diseño o gestión tales como:

— Editores de texto
— Ensambladores
— BASIC
— FORTRAN
— PLM
— PASCAL
— COBOL, etc.

Existen variantes de las memorias ROM, tales como la PROM, que son

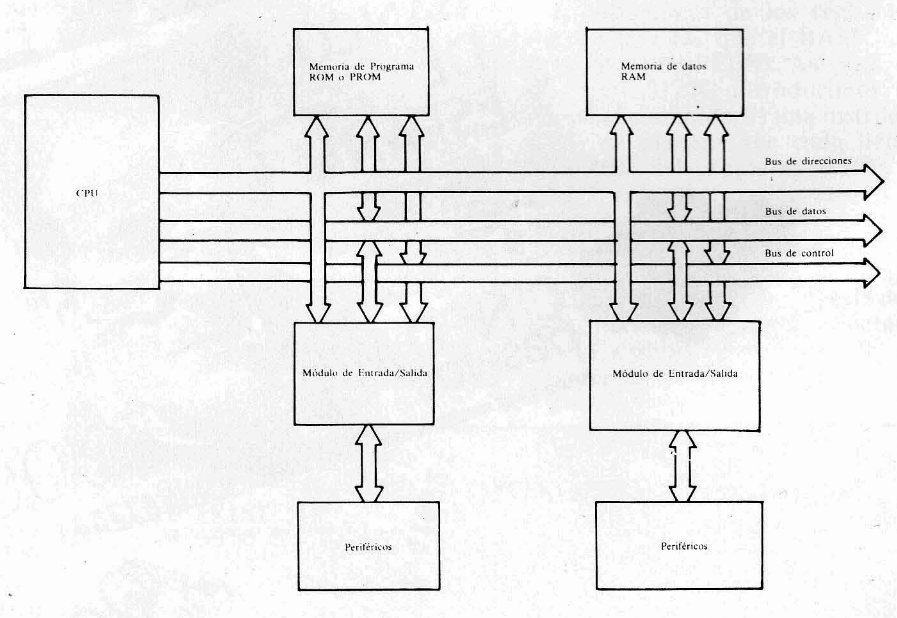


Figura 1. Diagrama de bloques

quina

Familia 6500

memorias programables por el usuario mediante grabadores especiales pero que una vez grabadas ya no se pueden regrabar con distinto contenido. Un error en el proceso de grabación las inutiliza.

EPROM y REPROM son memorias también programables por el usuario, pero con la gran ventaja de que se pueden borrar y ser grabadas de nuevo. El borrado se hace con radiaciones de rayos ultravioletas. Esta ventaja las hace muy apropiadas para la fabricación y desarrollo de prototipos.

El diagrama interno de los bloques que componen una memoria se puede ver en la figura 2.

Las memorias RAM pueden ser accedidas aleatoriamente, pudiendo leerse y escribirse en cualquier momento. Estas memorias son volátiles, es decir la información que tienen se pierde al desconectar la alimentación. Son utilizadas normalmente para guardar de una forma temporal, los datos que varían con el transcurso de las operaciones que ejecuta el sistema.

Por tanto podríamos decir que las memorias RAM están destinadas a contener los programas del usuario, o que el usuario introduce en el ordenador para ser ejecutados en un momento determinado. Los contenidos de todas las posiciones de la memoria RAM pueden ser grabados, leidos, visualizados y alterados, si fuere preciso, con la ayuda del programa MONITOR.

Las memorias RAM pueden ser estáticas o dinámicas. Las estáticas están configuradas en base a circuitos biestables de tipo D, con tecnología TTL (ver figura 3).

Las dinámicas se basan en la carga de un microscópico condensador, mediante la conducción o no de un transistor de tecnología MOS (ver fig. 4).

Por supuesto existen importantes ventajas a la hora de elegir entre una RAM dinámica o una memoria RAM estática. Pero estas no están exentas de ciertas desventajas que podemos enumerar.

Como ventajas se pueden destacar:

— Reducción de un tercio del coste por cada bit.

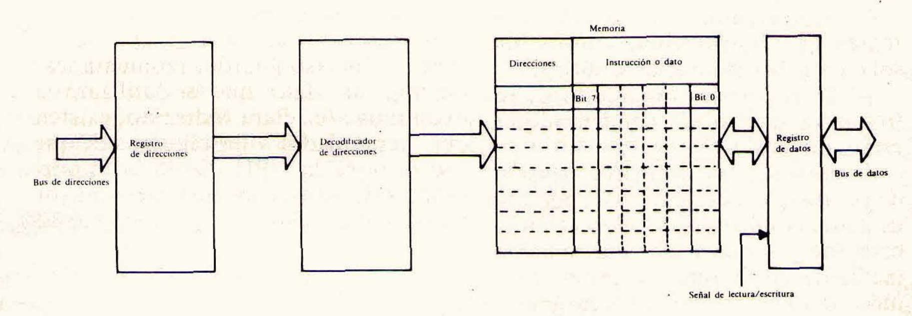


Figura 2. Diagrama interno de bloques de una memoria

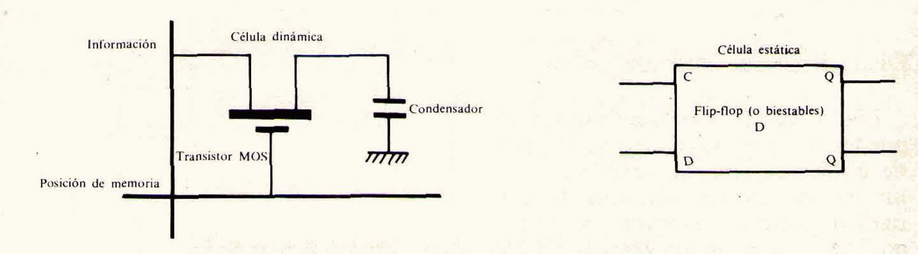


Figura 3. Células elementales de RAM. Estática y Dinámica

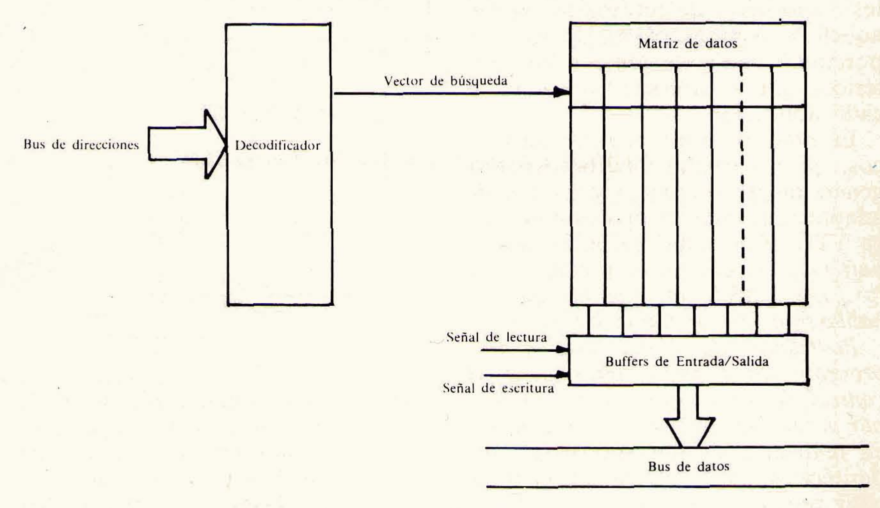


Figura 4. Constitución interna de una RAM

Familia 6500

— Se disipa del orden de cuatro veces menos en potencia (calor también).

— Aumenta la densidad de integración en chips, con respecto a las memorias estáticas.

— Se reduce el tiempo de acceso a más de la mitad del que se requiere en las memorias estáticas.

Como desventajas podríamos destacar las siguientes:

— Se necesitan tres tensiones diferentes de alimentación, contra una sola para las memorias estáticas.

— Es necesario un circuito de refresco ya que al ser condensadores, están sometidos a fugas.

Se entiende por refresco, el hecho de recargar el contenido de cada uno de estos condensadores con el fin de mantener, siempre su información, mediante un circuito que lee el contenido actual y lo vuelve a cargar.

La constitución interna de una memoria RAM es la que se ve en la figura 4.

Dispositivos de entrada-salida

Los dispositivos de entrada-salida pueden ser muy variados, yendo desde el simple teclado hexadecimal, a un teclado profesional, un display, una impresora, pantalla de video, *interface* para magnetófono, TTY, unidades de *diskette*, etc.

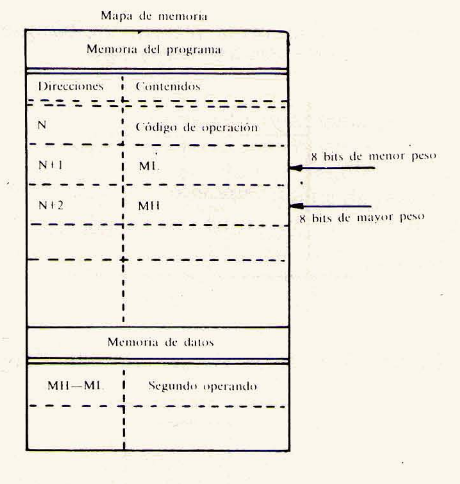
Por tanto podemos asegurar que los dispositivos de entrada-salida son aquellos circuitos destinados a proporcionar una conexión con los periféricos que se necesite controlar en cada aplicación.

El *interface*, o unión entre periféricos, y el sistema microordenador generalmente requieren circuitos de adaptación, implementados con lógica TTL, tales como decodificadores, *buffers, o con circuitos especiales como la PIA, UART, etc. de las que ya hablaremos en capítulos sucesivos.*

Podríamos decir que existen tres procedimientos importantes para el control de los periféricos. El primero por programa (software), que consiste en realizar consultas sucesivas a los periféricos para determinar cual de ellos requiere atención.

El segundo procedimiento, es por medio de interrupciones. En este caso es el periférico el que le dice a la unidad central de proceso que requiere su atención. Este procedimiento se estudiará exhaustivamente en capítulos sucesivos. Y el tercer procedimiento es el llamado Acceso Directo a Memoria (DMA), del que también hablaremos en su momento.

Para toda esta serie de operaciones necesitaremos utilizar continuamente un transvase de datos de un lado a otro, e incluso guardar momentaneamente un dato que se utilizará a continuación. Para todo esto, existen en Acumulador y los registros de que se dispone la CPU. Dada la importancia de éstos, conviene hacer mayor hincapié en ellos.



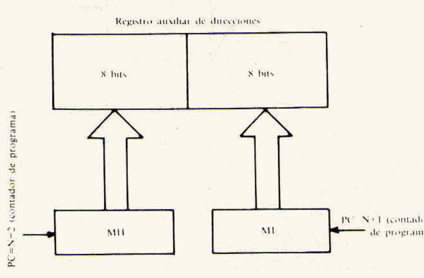


Figura 5. Así están organizados en la memoria los tres bytes que componen la instrucción ORA y como se carga, en dos fases, el registro auxiliar de direcciones

Registros disponibles en la CPU 6502

Como hemos dicho, mediante los registros se consigue la transferencia de datos entre el microprocesador y la memoria. En la figura 5 se puede ver la estructura de los registros de la CPU 6502, donde podemos distinguir el Acumulador, los registros X e Y, el contador de programa, etc.

El acumulador

Gracias a este registro (A) se realizan todas las comunicaciones de datos entre la unidad de proceso y la memoria. También se utiliza el Acumulador para el almacenamiento temporal de datos que están siendo transferidos de una posición de memoria a otra. Además, el Acumulador ayuda a realizar operaciones lógicas, comprobar el resultado de éstas, etc.

Registros X e Y

Son también llamados registros de índice o indexados, aunque se les conoce comúnmente como registros de aplicación general. Tienen la particularidad de poder incrementarse o decrementarse mediante simples operaciones apropiadas.

Cuando estudiemos el repertorio de instrucciones, veremos la utilidad de estos registros y las instruciones que los utilizan, dándonos cuenta de manera práctica de su verdadera utilidad.

Formas de realizar una instrucción

Poco a poco nos estamos acercando al conocimiento y utilización de las órdenes que tenemos que dar al ordenador, para que realice la tarea que le asignemos. Estas órdenes se denominan instrucciones. Antes de conocer el repertorio de instrucciones disponibles en esta familia, conozcamos como debe realizarse una instrucción.

Familia 6500

Toda instrución consta de dos fases bien definidas, que son:

— La búsqueda

— La ejecución

La búsqueda es común a todas las instrucciones y se inicia en el contador de programa, el cual contiene la dirección de memoria donde se encuentra el código correspondiente (código binario por supuesto) de la instrucción. Esta dirección es trasladada al registro de direcciones y desde allí a la memoria, mediante el *bus* de direcciones. Una vez que se decodifica en la memoria la posición que hemos direccionado, su contenido es trasladado por el *bus* de dátos e instruciones hasta los registros de datos e instrucciones, que están dentro de la CPU. En este preciso momento se da por acabada la fase de búsqueda y dará comienzo la fase de ejecución.

La característica más importante de la fase de búsqueda es ser común a todas y cada una de las instrucciones. Por el contrario la fase de ejecución dependerá de cada una de las instrucciones a tratar. La fase de ejecución comienza con el traslado del código de dicha instrucción hasta el decodificador de instrucciones, el cual se encargará de seleccionar cuales son las micro-instrucciones que se llevan a cabo siguiendo al secuenciador. Estas señales eléctricas que proporciona (secuencialmente) el secuenciador, prepara a la ALU (unidad aritmética-lógica) para realizar la operación correspondiente.

Las instrucciones pueden constar de uno, dos o tres bytes, según de cual se trate.

Para mejor entender este procedimiento veamos un ejemplo: supongamos la operación lógica ORA, que es la función lógica OR bit a bit entre el contenido del Acumulador y el de una posición determinada de la memoria, almacenándose el resultado final en el acumulador.

La instrucción que realiza esta operación consta de tres bytes, de los cuales el primero corresponde al código binario de la propia instrucción (el de la operación ORA), al segundo se le denomina ML., y el tercero es el byte de mayor peso de la dirección de memoria donde se encuentra el segundo operando y se denomina MH.

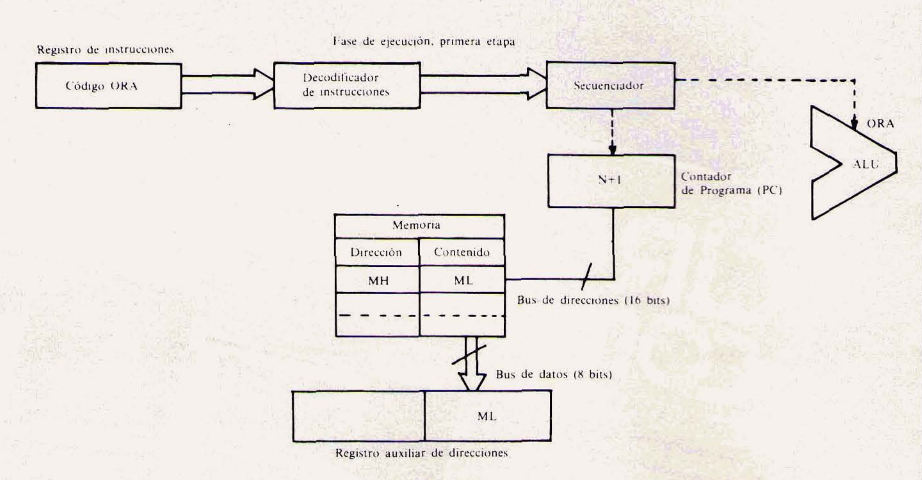


Figura 6.

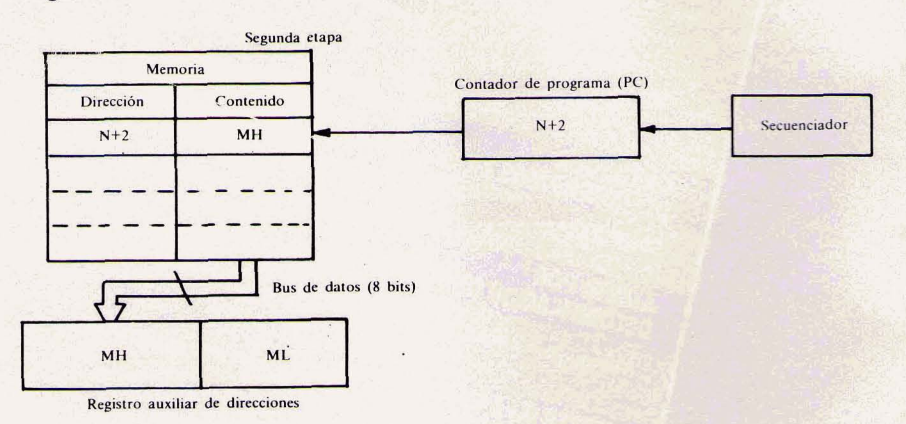


Figura 6 b.

La memoria de programa se dispondrá de la forma siguiente (ver fig. 5).

La figura 6 muestra gráficamente la forma en la que se lleva a cabo la fase de búsqueda de la instrucción ORA propuesta en nuestro ejemplo. En la figura 8 se ven los diagramas de las tres etapas que conforman la ejecución de la instrucción ORA.

En la primera etapa se ha cargado en el registro de instrucciones el código correspondiente a la función lógica OR. Este código es por supuesto la sucesión de unos y ceros (es un código binario) con el que se identifica la función OR. Los unos y ceros que forman el código entran en el decodificador de instrucciones y éste, como su nombre indica, decodifica dicho código y lo traduce en las señales eléctricas oportunas para controlar el secuenciador.

Como hemos visto, el secuenciador marca las pautas cual director de orquesta, para que cada señal entre en el momento oportuno.

El secuenciador, en nuestro ejemplo, da una señal a la ALU (unidad aritmético-lógica) indicándole que la operación a realizar es una operación lógica y que además se trata de la función OR. Por otro lado le manda otra señal al contador de programa, para que busque en la memoria donde se encuentra el byte menos significativo (ML), es decir, el de menor peso de la dirección de memoria, en cuyo contenido se encuentra el operando con el que vamos a hacer la función OR. Recuérdese que estamos haciendo la operación lógica OR

La revista imprescindible para todo el usuario de
Ordenadores COMMODORE

commodore Magazine

commodore Magazine

Año I - Número 1 - 200 ptas.

CALC RESULT, A FIN DE CUENTAS

Más potencia con Vistree

Cómo adaptar cualquier cassette

JUEGOS Y APLICACIONES PARA VIC-20 Y CBM 64

OFERTA ESPECIAL DE INTRODUCCION

Aproveche ahora esta irrepetible oportunidad para suscribirse a COMMODORE MAGAZINE. Envíe **HOY MISMO** la tarjeta adjunta, que no necesita sobre ni franqueo. Deposítela en el buzón más cercano. Inmediatamente recibirá su primer ejemplar de COMMODORE MAGAZINE y así durante un año (12 ejemplares).



Jerez, 3
Tel. 457 26 17
Madrid-16

Familia 6500

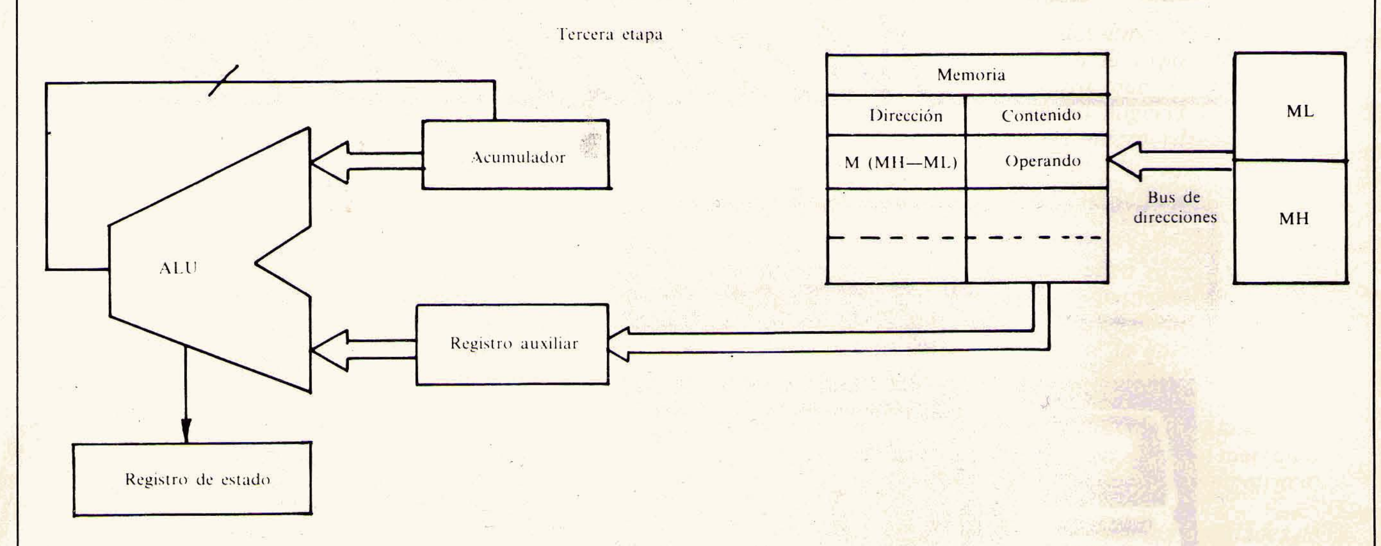


Figura 6 c

entre el contenido del acumulador y el contenido de una posición de memoria determinada.

Una vez buscado el byte de menor peso (ML) de la dirección (de la posición de memoria cuyo contenido es el dato buscado), éste se lleva a un registro auxiliar de direcciones a la espera del byte de mayor peso (MH).

En la segunda etapa, el contador de programa se incrementa y se busca el byte de mayor peso (MH) de la dirección de memoria, donde se encuentra el dato que estamos buscando. Para que sirva de refresco, conviene recordar que un bit es un estado lógico 0 ó 1 y un byte es el conjunto de 8 bits, por tanto ya podemos sospechar que la dirección de la posición de memoria consta de 16 bits, evidentemente ya que el *bus* de direcciones es de 16 bits.

Una vez conseguido el byte de mayor peso MH, se lleva al registro doble auxiliar de direcciones, donde nos está esperando el byte de menor peso ML.

Muy bien, ya conocemos en que dirección de la memoria se encuentra el dato buscado, para realizar la operación lógica OR con el contenido del Acumulador.

En la tercera etapa, a través del bus de direcciones llevamos a la memoria la dirección de la posición donde se encuentra el dato. El contenido de dicha posición de memoria (es decir el dato buscado) se transfiere a un registro auxiliar a la entrada de la ALU. En este momento la ALU, a través de este registro y del Acumulador, dispone de los dos datos con los que tiene que ejecutar la operación. La ALU ya conoce lo que ha de hacer (ver primera etapa) por tanto ejecuta la operación ORA y el resultado lo transfiere de nuevo al acumulador, por supuesto borrando el contenido anterior y ocupando este lugar el nuevo resultado obtenido.

El simple hecho de ejecutar esta instrucción hace que se modifique también el contenido del registro de estado, que como recordaremos nos va a proporcionar información a cerca de si la ejecución de la instrucción ha tenido acarreo, over flow, ha sido cero negativo, etc.

M. A. de Frutos

CIRCUITOS & COMPUTADORAS nº 52



PIDALA EN SU QUIOSCO

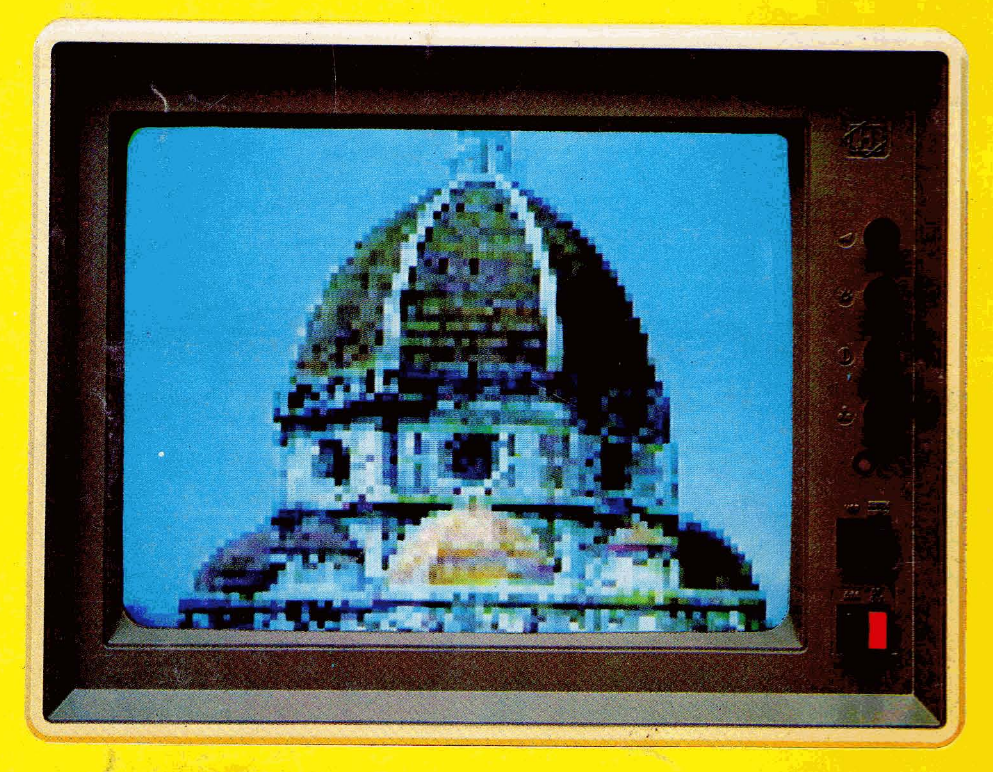
O SI LO PREFIERE, SUSCRIBASE LLAMANDO AL TELEFONO

(91) 457 26 17

HANTAREX ® Electronic Equipment Manufacturer

CALIDAD - RENTABILIDAD - SERVICIO

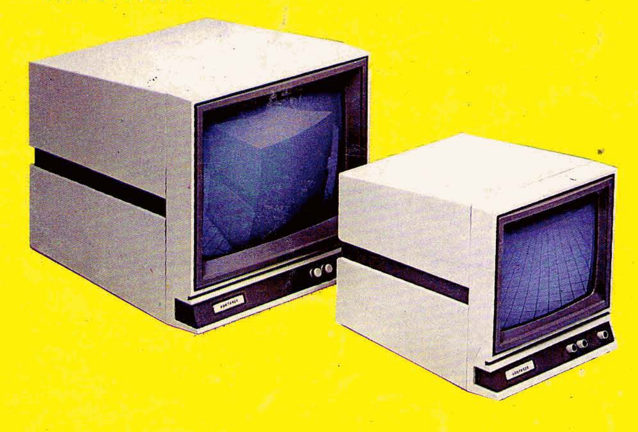
Aragón, 210, 1°, 1ª - Barcelona 11 - teléf. (93) 3232941 - telex 98017



Monitor color modelo CT 900/1-CT 900/2 14''



Serie Monitor monocromatico
modelo CTM 2000 9''/12''
modelo CT 2000 15''





Monitor color modelo CT 900 SR, MR, HR 14''

MONITORES

para todo tipo de ordenadores